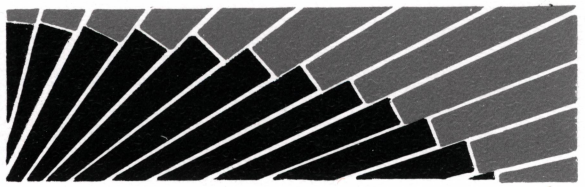
cobra

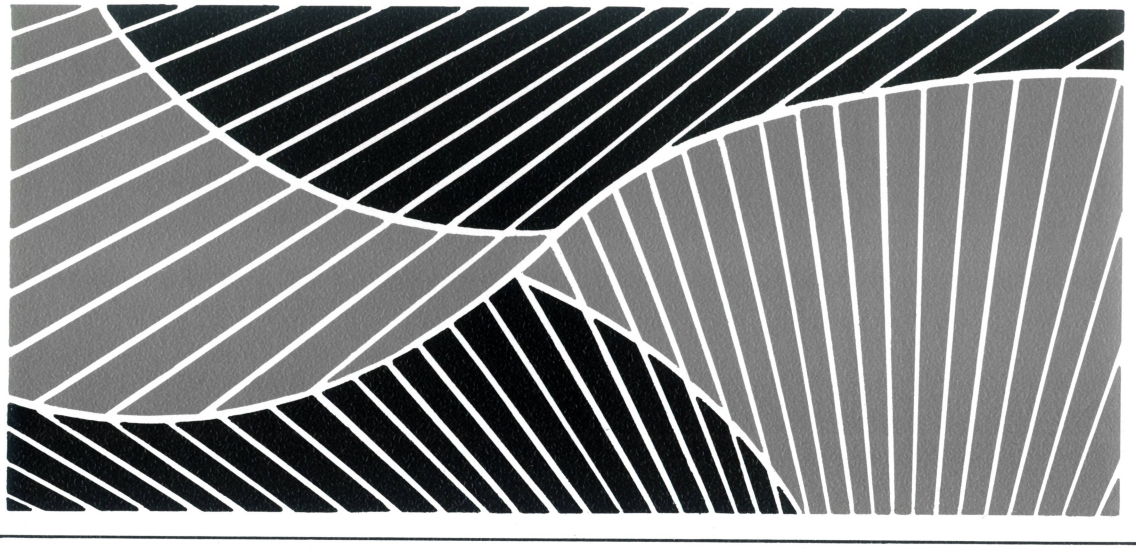


Manual de Referência

Comando SOX
VOLUME 2

SOX

cobra



SOX

Cobra

Computadores e Sistemas Brasileiros S/A

MANUAL DE REFERÊNCIA COMANDOS SOX

VOLUME 2

SX0189-03.0

MANUAL
DE REFERÊNCIA
COMANDOS SOX

3ª Edição

abril/89

SX0189-03.0

Este manual é uma publicação da Divisão de Testes e Integração de Sistemas / Setor de Documentação, aplicável a produtos comercializados pela Cobra.

O Fabricante reserva-se o direito de alterar a qualquer tempo, sem notificação prévia, as características e especificações técnicas do produto retratado nesta documentação, bem como promover, sempre que necessário, alterações e correções na forma e conteúdo desta publicação.

Cobra - Computadores e Sistemas Brasileiros S.A.

Todos os direitos reservados. Esta obra não pode ser comercializada, nem ter qualquer uso diferente do autorizado, nem pode também ser reproduzida, total ou parcialmente, por qualquer meio ou forma, sem prévio autorização escrita concedida pela Cobra.

Av. Comandante Guaranys, 447 - Jacarepaguá

22785 Rio de Janeiro RJ BRASIL

M4 **Comando - Desenvolvimento**

m4 - processador de macros.

SINTAXE

```
m4 [-s] [-Dmacro1[=val]] [-Umacro2] [arq... ]
```

DESCRIÇÃO

O comando **m4** é um processador de macros para ser usado como pré-processador para as linguagens C, Ratfor entre outras.

Cada arquivo é processado na ordem dada, se nenhum arquivo for especificado, ou se o nome de um arquivo for -, a entrada-padrão é assumida. O texto processado é gravado na saída-padrão.

OPÇÕES

-s

Desabilita a saída síncrona de linhas para o pré-processador C (i.e., diretiva do tipo #line). Esta opção deve ser a primeira a aparecer antes que qualquer nome de arquivo.

-D

Inicializa uma macro a ser utilizada no processamento.

-U

Cancela a definição de uma macro.

ARGUMENTOS

macro1

Nome da macro a ser inicializada.

val

Valor a ser atribuído a macro1. Se omitido é assumido igual a nulo.

macro2

Nome da macro a ser cancelada.

arq

Arquivos que contêm macros a serem avaliadas.

MACROS

As chamadas de macros seguem a seguinte sintaxe:

```
macro(arg1,arg2, ...,argn)
```

O caractere **(** (abre parêntese) deve seguir imediatamente o nome da macro. Caso o nome da macro definido não seja seguido por um caractere **(** (abre parêntese), será assumido que não tenha argumentos. Caracteres brancos, de tabulação e nova linha iniciais, são ignorados durante o processamento dos argumentos. Os nomes de macros são formados por uma seqüência de caracteres (letras alfabéticas, dígitos e sublinha "_"), sendo o primeiro caractere não dígito. Apóstrofos e acentos graves são usados para denotar cadeias (`...').

Quando o nome de uma macro é reconhecido, seus argumentos são delimitados pelos caracteres abre parêntese e fecha parentese. Os argumentos omitidos são substituídos por NULL. A avaliação de uma macro é feita durante a entrada dos argumentos. Uma macro contida em outra, ou seja, "aninhada", é tão eficaz quanto aquela contida no texto original. Após a entrada dos argumentos, o valor da macro é devolvido para a entrada, e o processamento continua.

O comando **M4** contém macros predefinidas. Elas podem ser redefinidas mas perdem assim, o significado original. Seus valores são nulos, a não ser que outro seja indicado nas descrições que seguem:

changecom[(arg1[,arg2])]

Troca os delimitadores de comentário da esquerda **(#)** e da direita (nova linha). Se não houver argumentos, o mecanismo do comentário é ignorado. Caso haja um argumento, o delimitador da esquerda será este, e o marcador da direita será o caractere de nova linha. Se houver dois argumentos, ambos serão mudados. Os delimitadores de comentário podem conter até cinco caracteres.

changequote(arg1,arg2)

Troca os caracteres delimitadores aspas simples pelo primeiro e segundo argumentos. Os símbolos podem conter até cinco caracteres. Se usado sem argumentos restabelece os valores originais (i.e.,``'').

decr(arg)

Retorna o valor de seu argumento decrementado de 1 (um).

define(arg1,arg2)

O primeiro argumento fornece o nome da macro e o segundo o valor. Cada ocorrência de **$n** em arg2 (onde n é um dígito) é substituido pelo n-ésimo argumento. O argumento 0 (zero) é o nome da macro, argumentos ausentes são substituídos pela cadeia nula; **$#** é substituído pelo número de argumentos; **$*** é substituído pela lista de todos os argumentos separados por vírgula; e **$@** é semelhante a **$***.

defn(arg1,arg2,...,argn)

Retorna a definição de seu(s) argumento(s) entre aspas simples. É útil para alterar o nome de macros especialmente as macros predefinidas.

divert(arg)

O comando **m4** manipula dez arquivos de saída, numerados de 0 a 9. A saída final é a concatenação dos arquivos em ordem numérica; inicialmente o arquivo 0 (zero) é o arquivo corrente. Esta macro troca o arquivo de saída corrente pelo indicado no argumento. Uma saída alterada para um arquivo fora dos numerados 0 a 9 é desprezada.

divnum

Retorna o valor da saída corrente.

dnl

Lê e ignora todos os caracteres até o último incluindo o caractere nova linha.

dumpdef[(arg1,arg2,...,argn)]

Imprime os nomes e definições de macros correntes para os itens dados como argumentos, ou todos os itens, se não houver argumento.

errprint(arg)

Imprime seu argumento na saída-padrão de erro (stderr).

eval(arg1[,arg2[,arg3]])

Avalia o primeiro argumento como uma expressão aritmética, usando aritmética de 32 bits. Os operadores permitidos são: +, -, *, /, %, ** (exponenciação), &, |, ^ e ~(bit-a-bit); relacionais; parênteses. Números em octal e hexadecimal possuem sintaxe idêntica a da linguagem C. O segundo argumento especifica a base do resultado, o default é 10. O terceiro argumento pode ser usado para especificar o número mínimo de dígitos no resultado.

ifdef(arg1,arg2[,arg3])

Se o primeiro argumento for definido, o valor retornado é o segundo argumento caso contrário o terceiro. Caso não exista o terceiro argumento, o valor nulo será retornado.

ifelse(arg1,arg2,arg3,...argn)

Tem três ou mais argumentos. Se o primeiro argumento for uma cadeia igual à do segundo argumento, então o valor é o terceiro argumento. Do contrário, e caso haja mais de quatro argumentos, o processo é repetido com os argumentos 4, 5, 6 e 7. Caso contrário, o valor é a quarta cadeia ou, o valor nulo, se esta não existir.

incr(arg)

Retorna o valor de seu argumento incrementando de 1 (um). O valor do argumento é obtido pela interpretação de uma cadeia de digitos como um número decimal.

include(arg)

Retorna o conteúdo do arquivo indicado pelo argumento.

index(arg1[,arg2])

Retorna a posição onde o segundo argumento começa dentro do primeiro argumento (origem = zero), ou -1 se o segundo argumento não ocorrer no primeiro.

len(arg)

Retorna o número de caracteres presentes no seu argumento.

m4exit (arg)

Saída imediata do **m4**. O argumento, dado é o código de saída. O valor default é 0 (zero).

m4wrap(arg)

O argumento será devolvido à entrada quando o EOF final for encontrado.

Exemplo: **m4wrap** (`cleanup()').

maketemp(arg)

Preenche uma cadeia XXXXX presente no argumento com a identificação do processo corrente (vide syscmd).

popdef(arg1,arg2,...,argn)

Remove a definição corrente de seu(s) argumento(s), e restabelece a definição anterior, caso exista.

pushdef(arg0,arg1)

Semelhante ao **define**, mas salva a definição anterior.

shift(arg1,arg2,...,argn)

Retorna todos os seus argumetos entre aspas simples, separados por vírgulas, exceto o primeiro. Os caracteres aspas simples anulam o efeito de uma nova leitura.

sinclude(arg)

É idêntico ao **include**, mas não imprime advertência caso o arquivo não possa ser acessado.

substr(arg1,arg2[,arg3])

Retorna uma subcadeia do seu primeiro argumento. O segundo argumento é um número (origem = zero) selecionando o primeiro caractere; o terceiro caractere indica o tamanho da subcadeia. Se o terceiro argumento for omitido, a subcadeia estende-se até o final da primeira.

syscmd(arg)

Executa o comando do sistema dado pelo argumento. Nenhum valor é retornado.

sysval

É o código de retorno da última chamada feita por **syscmd.**

traceoff[(arg1,arg2,...,argn)]

As macros cujo rastreamento foi ligado através de **traceon,** só podem ter o rastro desligado através de **traceoff.**

traceon[(arg1,arg2,...,argn)]

Sem argumento, ativa o rastreamento para todas as macros (incluindo as macros predefinidas). Caso contrário, rastreia as macros dadas como argumentos.

translit(arg1,arg2,arg3)

Substitui no primeiro argumento os caracteres iguais aos do segundo pelos do terceiro (correspondente). Não são permitidas abreviações. Exemplo: translit(cobra,aeiou,12345) resultaria c4br1.

undefine(arg)

Remove a definição da macro dada como argumento.

undivert[(arg1,arg2,...,argn)]

Causa a saída imediata do texto dos arquivos indicados pelos argumentos, ou todos os arquivos (0 a 9 , ver **divert**), se não houver argumento. O texto assim afetado pode ser desviado para outro arquivo. Esta operação descarta o texto desviado.

DIÁLOGO

Não existe.

CÓDIGOS DE RETORNO

zero - término sem erro.

diferente de zero - término com erro.

ARQUIVOS UTILIZADOS

Não existem.

VEJA TAMBÉM

cc (comando)
cpp (comando)

EXEMPLOS

```
. m4
  define(N,5)
  main()
  {
     if(N>2)
       printf("OK\n");
  }
  ^D
  main()
  {
     if(5>2)
       printf("OK\n");
  }
```

MAKE **Comando - Desenvolvimento**

make - mantém, atualiza e gera grupos de programas.

SINTAXE

```
make [-f makefile] [-p] [-i] [-k] [-s] [-r]
                   [-n] [-e] [-t] [-q] [nome...]
```

DESCRIÇÃO

O comando **make** aceita uma especificação das interdependências dos vários módulos de um programa. Baseado nas datas de modificação dos arquivos e nas especificações que aparecem no arquivo de descrição **(makefile)**, mostrando as relações de dependência entre os vários módulos, **make** deduz quais os módulos que estão desatualizados e automaticamente executa as linhas de comando presente em **makefile**, que geralmente representam ações para atualizar os módulos.

Como exemplo, consideremos um módulo de programa principal armazenado no arquivo chamado teste.c e que utiliza outras rotinas contidas nos arquivos testel.c e teste2.c. Iremos também supor que todas as rotinas façam uso do arquivo de inclusão teste.h. A figura a seguir mostra as relações de interdependência entre os vários módulos.

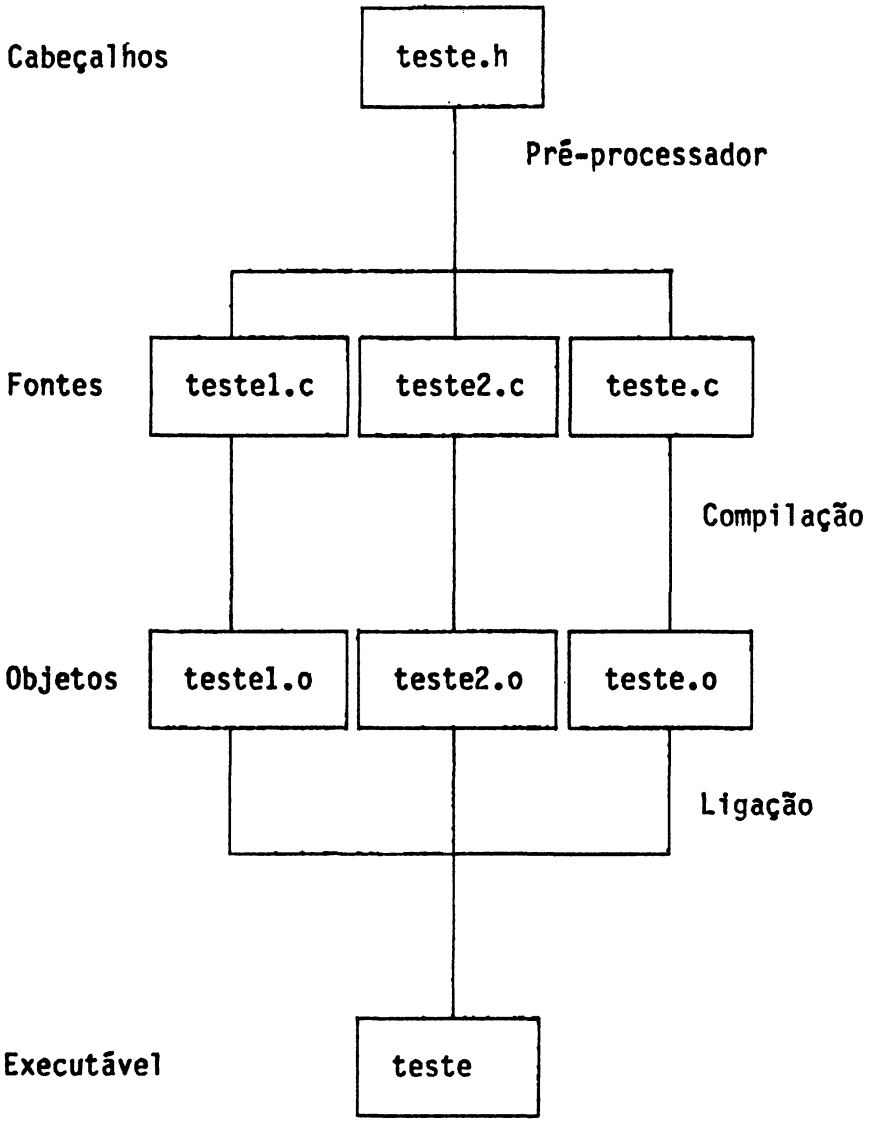

Cabeçalhos
teste.h
Pré-processador
Fontes
teste1.c
teste2.c
teste.c
Compilação
Objetos
teste1.o
teste2.o
teste.o
Ligação
Executável
teste

Figura 1

Como podemos ver pela Figura 1, o arquivo-objeto testel.o é criado através da compilação do texto em testel.c incluindo teste.h. Assim, em **makefile** poderíamos colocar uma linha de dependências:

```
testel.o : testel.c teste.h
```

Após a linha de dependências **makefile** deverá incluir nas próximas linhas comandos que atualizarão o módulo. O primeiro caractere deverá ser o de tabulação. Portanto toda a especificação para as dependências e recriação do módulo-objeto testel.o é:

```
testel.o : testel.c teste.h
        cc -c testel.c
```

Para teste2.o e teste.c o mesmo procede, e teríamos no arquivo de descrição as linhas:

```
teste2.o : teste2.c teste.h
        cc -c teste2.c
teste.o  : teste.c teste.h
        cc -c teste.c
```

A opção **-c** do comando **cc** direciona-o a executar uma compilação parcial do arquivo-fonte (**.c**) produzindo um arquivo-objeto.

O programa teste depende de três módulos-objeto que são testel.o, teste2.o e teste.o. Assim, encontraríamos em **makefile**:

```
teste : testel.o teste2.o teste.o
        cc -o teste testel.o teste2.o teste.o
```

A opção **-o** para o comando **cc** instrui o ligador a colocar a saída executável no arquivo teste.

Com isso poderíamos ter em um arquivo **makefile** chamado arqmake as seguintes linhas:

```
teste : teste1.o teste2.o teste.o
        cc -o teste teste1.o teste2.o teste.o
teste1.o : teste1.c teste.h
        cc -c teste1.c
teste2.o : teste2.c teste.h
        cc -c teste2.c
teste.o : teste.c teste.h
        cc -c teste.c
```

Este arquivo poderia então ser utilizado para recompilar os módulos apropriados cada vez que um módulo de inclusão ou fonte fosse alterado. Para isso bastaria que executássemos

```
make -f arqmake teste
```

O objetivo do comando **make** dado acima é ver se o arquivo teste está desatualizado e, se estiver, criar uma nova versão. Um módulo estará desatualizado se depender de algum módulo que seja mais recente que ele mesmo.

Por exemplo se o arquivo de inclusão teste.h fosse alterado, o comando acima executaria:

```
cc -c teste1.c
cc -c teste2.c
cc -c teste.c
cc -o teste teste1.o teste2.o teste.o
```

O produto final é uma nova versão de teste.

Se agora uma nova alteração for feita em testel.c teríamos:

```
cc -c testel.c
cc -o teste testel.c teste2.c teste3.c
```

O mesmo **makefile** poderia ser usado para gerar apenas uma nova versão de testel.o, designando-o como argumento:

```
make -f arqmake testel.o
```

gerando apenas o primeiro comando do exemplo anterior.

Os seguintes nomes especiais de módulos-alvo podem ser definidos no **makefile** e são interpretados como se segue:

.DEFAULT - Se um arquivo deve ser gerado mas não há comandos explícitos, nem regras internas relevantes, os comandos relacionados como o nome .DEFAULT são usados, se essa entrada existir.

.PRECIOUS - Dependentes deste módulo-alvo não serão removidos quando um sinal SIGQUIT ou SIGINT ocorrer.

.SILENT - O mesmo efeito da opção **-s**.

.IGNORE - O mesmo efeito da opção **-i**.

No caso geral, o comando **make** analisa as dependências definidas no **makefile** e gera comandos para atualizar um ou mais nomes de módulos-alvo (nomes-alvo). O argumento **nome** é tipicamente um programa e, se nenhum for fornecido no comando, **make** tenta gerar o primeiro nome-alvo definido no **makefile**. Se nenhuma opção **-f** encontra-se presente, **make** tenta examinar os arquivos **makefile**, **Makefile** e os arquivos SCCS **s.makefile** e **s.Makefile**, nesta ordem. Se **makefile** é - (hífen) a entrada padrão é usada. Mais de um par **-f makefile** pode aparecer no comando.

Um **makefile** consiste em uma seqüência de entradas que especificam dependências. A primeira linha de uma entrada é uma lista não-vazia de nomes-alvo, separados por brancos, seguida de : (dois pontos) e seguida de uma lista (eventualmente vazia) de arquivos pré-requisitos, ou dependências. O texto que segue a um ; (ponto-e-vírgula) e todas as linhas seguintes que comecem com uma tabulação (ASCII HT), são comandos a serem executados para atualizar os nomes-alvo. Uma linha que comece com # (cardinal) é considerada um comentário.

A primeira linha que não começar com tabulação ou inicia uma nova dependência, ou uma definição de macro (ver Macros). A seqüência <contra-barra><avanço de linha> (LF) pode ser usada para continuar um comando por mais de uma linha. Tudo que é impresso por **make**, exceto a tabulação inicial, é enviada como encontrado diretamente para o interpretador de comandos.

O comando **make** atualiza um nome-alvo apenas se seus dependentes forem mais recentes que o nome-alvo. Todos os seus arquivos pré-requisitos são recursivamente adicionados à linha de nomes-alvo. Arquivos não encontrados são tratados como desatualizados. As linhas de comando são executadas uma por vez. Os primeiros um ou dois caracteres de um comando podem ser: -, @ , -@, ou @ -. Se @ estiver presente, a impressão do comando é suprimida. Se - estiver presente, **make** ignora um erro. Uma linha de comando é impressa quando é executada, a menos que a opção **-s** tenha sido especificada na ativação do comando, ou a entrada **.SILENT** esteja no **makefile**, ou um dos dois primeiros caracteres da linha seja @. A opção **-n** especifica impressão sem execução. Contudo, se a linha de comando possuir a cadeia $(**MAKE**), a linha é sempre executada. A opção **-t** ("touch") atualiza a data de modificação de um arquivo sem executar qualquer comando.

Comandos que retornam códigos "diferente de zero" normalmente terminam **make**. Se a opção **-i** estiver presente, ou a entrada **.IGNORE**, aparecer no **makefile**, ou se linha de comando contiver - (hífen) na sua seqüência inicial, o erro é ignorado. Se a opção **-k** estiver presente, o trabalho é abandonado na entrada corrente, mas continua em outros ramos que não dependam daquela entrada.

A ocorrência de sinais SIGINT e SIGQUIT fazem com que o módulo-alvo seja removido, a menos que ele seja um dependente do nome especial .PRECIOUS.

Ambiente

O ambiente é lido por **make.** Todas as suas variáveis são assumidas como sendo definições de macros.e tratadas como tais. As variáveis do ambiente são processadas antes de qualquer **makefile** e após as regras internas; em conseqüência, as atribuições de macros em um **makefile** se sobrepõem às variáveis ambientais. A opção **-e** causa o efeito inverso.

A variável ambiental MAKEFLAGS é processada por **make** como contendo qualquer opção de entrada legal (exceto **-f** e **-p)** definida para a linha de comando. Além disso, ao ser chamado, **make** "inventa" a variável (se esta não estiver no ambiente), coloca nesta as opções correntes e passa-a em chamadas de comandos. Sendo assim, MAKEFLAGS sempre contém as opções de entrada, correntes. Este fato é muito interressante para supermakes (isto é, make, que executa **make**). De fato, como notado acima, quando a opção **-n** é utilizada, o comando $(**MAKE**) é executado de qualquer maneira; então, alguém pode executar **make -n** recursivamente em um sistema de software completo para ver o que seria executado. Isto é devido ao fato da opção -n ser colocada em MAKEFLAGS e passada para chamadas futuras de $(**MAKE**). Este é um modo de se depurar todos os "make files" de um projeto de software, sem nada executar realmente.

Macros

Entradas da forma **cadeia1 = cadeia2** são definições de macro. A macro **cadeia2** é definida como todos os caracteres até um caractere de comentário, ou de avanço de linha sem escape. Subseqüentes aparecimento de:

$(cadeia1 [:subst1= subst2]])

são trocados por **cadeia2.** Os parênteses são opcionais se um nome de macro de um caractere for usado, e não existir nenhuma seqüência de substituição. A seqüência opcional **:subst1=subst2** é uma seqüência de substituição. Se estiver especificada, todas as ocorrências não sobrepostas de **subst1** na macro sendo tratada são substituídas por **subst2.** Cadeias (para o propósito deste tipo de substituição) são delimitadas por brancos, tabulações, caracteres de avanço de linha e começo de linha. Um exemplo de uso da seqüência de substituição é mostrado no item **Bibliotecas.**

Macros Internas

Existem cinco macros definidas internamente que são úteis na definição de regras para a geração de módulos-alvo.

$*** - A macro **$* significa a parte do nome do arquivo do atual dependente, com o sufixo removido. É avaliada apenas para regras de inferência.

$@** - A macro **$ @ significa o nome completo do módulo-alvo corrente. É avaliada apenas para dependências definidas explicitamente.

$<** - A macro **$< é avaliada apenas para regras de inferência e para a regra **.DEFAULT**. Representa o módulo que está desatualizado com respeito ao módulo-alvo. Assim, na regra **.c.o**, a macro **$<** seria substituída pelo arquivo **.c**. Um exemplo de construção de arquivos **.o** otimizados a partir de arquivos **.c** é:

```
.c.o:
cc -c -O $*.c
```

ou:

```
.c.o:
cc -c -O $<
```

$?** - A macro **$? é avaliada quando regras explícitas provenientes do **makefile** são avaliadas. Representa a lista de pré-requisitos que estão desatualizados com relação ao módulo-alvo; essencialmente, os módulos que devem ser reconstruídos.

$% - A macro **$%** é avaliada apenas quando o módulo-alvo for um membro de um arquivo biblioteca, da forma bib(arq.o). Neste caso, **$@** seria substituído por bib e **$%** membro da biblioteca, arq.o.

Quatro das cinco macros podem ter formas alternativas. Quando um **D** ou **F** maiúsculos forem concatenados com qualquer uma dessas quatro macros, o significado é alterado para "a parte do diretório", para **D**, e "a parte do arquivo", para **F**. Assim, **$(@D)** refere-se a parte do diretório da cadeia **$@**. Se não há parte referente a um diretório, **./** é gerado. A única macro excluída desta forma alternativa é **$?**.

Regras de Inferência

Certos nomes (por exemplo, aqueles terminados em **.o**) têm pré-requisitos inferíveis, tais como **.c**, **.s**, etc. Se nenhum comando de atualização para tal arquivo aparece no **makefile**, e se um pré-requisito inferível existe, este pré-requisito é compilado para gerar o módulo-alvo. Neste caso, **make** tem regras de inferência que permitem a contrução de arquivos a partir de outros arquivos pelo exame de sufixos e pela determinação da regra de inferência apropriada a ser usada. Regras de inferência no **makefile** anulam regras default.

As regras internas para o **make** são compiladas dentro do programa **make**. A lista de regras default é a seguinte:

```
.h~.h:
        $(GET) $(GFLAGS) -p $< >$*.h

.s~.a:
        $(GET) $(GFLAGS) -p $< >$*.s
        $(AS) $(ASFLAGS) -o $*.o$*.s
        ar -rv $@ $*.o
        -rm -f $*.[so]

.c~.a:
        $(GET) $(GFLAGS) -p $< >$*.c
        $(CC) -c $(CFLAGS) &<
        ar -rv $@ $*.o
        -rm -f $*.[co]

.c.a:
        $(CC) -c $(CFLAGS) $<
        ar -rv $@ $*.o
        -rm -f $*.o
```

```
.l.c:
        $(LEX) $<
        mv lex.yy.c $@

.y~.c:
        $(GET) $(GFLAGS) -p $< > $*.y
        $(YACC) $(YFLAGS) $*.y
        mv y.tab.c $*.c
        -rm -f $*.y

.y.c:
        $(YACC) $(YFLAGS) $<
        mv y.tab.c $@

.l~.o:
        $(GET) $(GFLAGS) -p $< > $*.l
        $(LEX) $(LFLAGS) $*.l
        $(CC) $(CFLAGS) -c lex.yy.c
        rm -f lex.yy.c $*.l
        mv lex.yy.o $*.o

.l.o:
        $(LEX) $(LFLAGS) $<
        $(CC) -c $(CFLAGS) lex.yy.c
        rm lex.yy.c
        mv lex.yy.o $@

.y~.o:
        $(GET) $(GFLAGS) -p $< > $*.y
        $(YACC) $(YFLAGS) $*.y
        $(CC) (CFLAGS) -c y.tab.c
        rm -f y.tab.c $*.y
        mv y.tab.o $*.o
```

```
.y.o:
        $(YACC) $(YFLAGS) $<
        $(CC) $(CFLAGS) -c y.tab.c
        rm y.tab.c
        mv y.tab.o $@

.s~.o:
        $(GET) $(GFLAGS) -p $< > $*.s
        $(AS) $(ASFLAGS) -o $*.o $*.s
        -rm -f $*.s

.s.o:
        $(AS) $(ASFLAGS) -o $@ $<

.c~.c:
        $(GET) $(GFLAGS) -p $< >$*.c

.c~.o:
        $(GET) $(GFLAGS) -p $< >$*.c
        $(CC) $(CFLAGS) -c $*.c
        -rm -f $*.c

.c.o:
        $(CC)$(CFLAGS) -c $<

.sh~:
        $(GET) $(GFLAGS) -p $< >$*.sh
        cp $*.sh $*; chmod 0777 $@
        -rm -f $*.sh

.sh:
        cp $< $@; chmod 0777 $@
```

```
.c~:
        $(GET) $(GFLAGS) -p $< >$*.c
        $(CC) $(CFLAGS) $(LDFLAGS) $*.c -o $*
        -rm -f $*.c

.c:
        $(CC) $(CFLAGS) $(LDFLAGS) $< -o $@
```

Um til nas regras acima se refere a um arquivo SCCS. Assim, a regra **.c~.o** transformaria um arquivo-fonte C SCCS em um arquivo-objeto (**.o**). Por ser o **s.** de um arquivo SCCS um prefixo , ele é incompatível com o ponto de vista de sufixos do **make**. Sendo assim, o til é um modo de mudar qualquer referência a arquivo em uma referências a arquivo SCCS.

Uma regra com apenas um sufixo (por exemplo, **.c:**) é a definição de como construir **x** de **x.c.** Com efeito, o outro sufixo é nulo. Isto é interessante para se construir módulos-alvo de apenas um arquivo-fonte (por exemplo, programas C simples, arquivos de comandos, etc.).

.SUFFIXES

A ordem de sufixos nessa lista é relevante: o primeiro nome possível para o qual exista tanto um arquivo como uma regra é inferido como um pré-requisito. A lista default é :

```
.SUFFIXES :
   .o .c .c~ .y .y~ .l .l~ .s .s~ .sh .sh~ .h .h~ .
```

O comando make -pf - 2>/dev/null< /dev/null exibirá as regras internas, bem como a lista de sufixos implementados na máquina utilizada. Múltiplas listas de sufixo se acumulam; .SUFFIXES: sem nenhuma dependência limpa a lista de sufixos.

Regras de Inferência

O primeiro exemplo poderia ter sido feito mais resumidamente como se segue:

```
teste: teste1.o teste2.o teste.o
    cc -o teste teste1.o teste2.o teste.o
teste1.o teste2.o teste.o: teste.h
```

Isto é devido ao fato de **make** possuir um conjunto de regras internas para a construção de arquivos. O usuário pode adicionar regras a esta lista simplesmente colocando-as no **makefile.** Certas macros são usadas nas regras de inferência default para permitir a inclusão de matéria opcional em qualquer comando resultante. Por exemplo, BCFLAGS, BLFLAGS e BYFLAGS são usadas como opções de compilação para **cc, lex** e **yacc,** respectivamente. As macros default são as seguintes:

```
GET=get
AS=as
CC=cc
LD=ld
LEX=lex
YACC=yacc
MAKE=make
ASFLAGS=
CFLAGS=
LDFLAGS=
LFLAGS=
YFLAGS=
MAKEFLAGS=
GFLAGS=
```

A inferência de pré-requisitos pode ser controlada. A regra para se criar um arquivo com sufixo **.o** de um arquivo com sufixo **.c** é especificada com uma entrada com **.c.o:** como módulo-alvo e sem dependentes. Os comandos associados com o módulo-alvo definem a regra para criar um arquivo **.o** de um arquivo **.c.** Qualquer módulo-alvo que não tenha barras e comece com um ponto é identificado como uma regra, e não como um verdadeiro módulo-alvo.

Bibliotecas

Se um módulo-alvo ou um nome de dependência contém parênteses, é assumido como sendo um arquivo biblioteca, com a cadeia entre parênteses referindo-se a um membro da biblioteca. Então **bib(arq.o)** e **$(BIB)(arq.o)** se referem a uma biblioteca que contém **arq.o.** (Supõe-se que a macro BIB tenha sido previamente definida).

A expressão **$(BIB)(arquivo1.o arquivo2.o)** não é legal. Regras relacionadas com bibliotecas têm a forma **.XX.a**, onde **XX** é o sufixo a partir do qual o membro deve ser construído. O uso mais comum da interface para bibliotecas segue abaixo. Aqui, assumiu-se que todos os arquivos-fonte são fontes em C:

```
bib: bib(arq1.o) bib(arq2.o) bib(arq3.o)
     echo bib agora esta atualizada
.c.a:
    $(CC) -c $(CFLAGS) $<
    ar-rv $@ $*.o
    rm -f $*.o
```

De fato, a regra **.c.a** listada acima é construída dentro do **make** e é desnecessária neste exemplo. Segue-se um exemplo mais interessante, mas mais limitadoe:

```
bib: bib(arq1.o) bib(arq2.o) bib(arq3.o)
    $(CC) -c $(CFLAGS) $(?:.o=.c)
    ar-rv bib $?
    rm $?;@ echo bib agora esta atualizada
.c.a:;
```

Aqui, o modo de substituição de expansões de macro é utilizado. A lista **$?** é definida como sendo o conjunto de nomes de arquivos-objeto (dentro de **bib**) cujos arquivos-fonte C estão desatualizados. O modo de substituição transforma **.o** para **.c.** Note também a desabilitação da regra **.c.a:**, que teria criado cada arquivo objeto, um por um. Esta particular construção aumenta consideravelmente a velocidade de manutenção de bibliotecas. Este tipo de construção torna-se muito incômodo se a biblioteca contém um misto de arquivos em assembler e programas em C.

OPÇÕES

-f makefile

Nome do arquivo de descrição. O argumento **makefile** é assumido como sendo o nome de um arquivo de descrição. O nome de arquivo - denota a entrada padrão.

-p

Imprime o conjunto completo de definições de macro e as descrições para geração dos arquivos (módulos-alvo).

-i

Ignora códigos retornados pelos comandos executados. Esse modo também é executado se o falso módulo-alvo .IGNORE aparecer no arquivo de descrição.

-k

Abandona o trabalho na entrada corrente se esta fracassar, mas continua em outros ramos que não dependam desta entrada.

-s

Modo silencioso. Não imprime linhas de comando antes de executá-las. Esse modo também é executado se o falso módulo-alvo **.SILENT** aparecer no arquivo de descrição.

-r

Não usa regras internas predefinidas.

-n

Modo de não-execução de comandos. Imprime comandos, mas não os executa. Mesmo linhas começando por @ são impressas. Entretanto, comandos com $(MAKE) são sempre executados.

-e

Variáveis do ambiente se sobrepõem a atribuições dentro dos arquivos de descrição.

-t

Altera a data de modificação dos nomes-alvo, fazendo com que módulos-alvo fiquem atualizados, em vez de emitir os comandos do **makefile.**

-q

O comando **make** retorna um código zero ou diferente de zero, dependendo se o módulo-alvo está ou não atualizado.

ARGUMENTO

nome

Nome-alvo a ser gerado.

DIÁLOGO

Não existe.

CÓDIGOS DE RETORNO

zero - término sem erro.

diferente de zero - término com erro.

ARQUIVOS UTILIZADOS

makefile
Makefile
s.makefile
s.Makefile

VEJA TAMBÉM

cc (comando)
lex (comando)
sh (comando)
yacc (comando)

Página intencionalmente em branco.

MANUT **Comando - Administração**

manut - verifica/consiste arquivo com organização seqüencial indexada.

SINTAXE

```
manut [-ynips] arq [indice]
```

DESCRIÇÃO

O comando **manut** verifica a integridade de um arquivo com organização seqüencial indexada e recupera, quando for possível, as estruturas de dados defeituosas em virtude de erros causados por queda de força. Deve ser ativado sempre que o arquivo estiver marcado como inconsistente, de modo a possibilitar sua utilização. O usuário ainda pode ativar o **manut** a qualquer momento desde que nenhum outro usuário esteja com o arquivo aberto, caso contrário, o **manut** será cancelado.

Se não existem opções, o **manut** faz a verificação de todos os índices definidos para o arquivo. Neste caso, se alguma inconsistência for encontrada e o arquivo foi criado com o atributo de gravação com sincronismo (ver "Manual de Referência Sistema de Arquivamento Estendido SOX" (BX01)), o comando procura resolver essa inconsistência a revelia do usuário, caso contrário, isto é, o arquivo tiver sido criado sem atributo de gravação com sincronismo, o utilitário resolve a inconsistência por reconstrução ou eliminação do índice, com a intervenção do usuário.

OPÇÕES

-y

Assume a resposta "sim" a todas as perguntas feitas pelo **manut.**

-n

Assume a resposta "não" a todas as perguntas feitas pelo **manut.**

-i

Usuário não deseja verificar/consistir todos os índices. Os índices a serem verificados são passados após o argumento **arg.**

-p

A consistência do arquivo é feita com o conhecimento do usuário, que decide pela reconstrução ou eliminação dos índices, caso seja encontrada uma inconsistência em suas estruturas ou quando de inconsistência em página de dados de um arquivo sem índice definido, sobre a eliminação ou não desta página do arquivo.

-s

Ignora tentativas de cancelamento da manutenção através das teclas de controle do terminal. Neste caso, o cancelamento só é possível via chamada kill. Caso esta opção não seja especificada e o usuário digitar alguma tecla de controle, o **manut** solicita a confirmação do cancelamento, com a seguinte mensagem:

Quer Interromper a Manutenção (s/n)

ARGUMENTOS

arq

Nome do arquivo a ser verificado/consistido.

índices

Índices a serem verificados/ consistidos separados por vírgulas. Só válido se a opção **-i** presente.

DIÁLOGO

As mensagens, diálogos e procedimentos para tratamento de erros do comando variam de acordo com as características do arquivo e são descritos a seguir.

1. DADO DIRETO

** Fase 1 - verificando índice

Nesta fase, é feita a verificação da estrutura de dados do índice. Se alguma inconsistência for encontrada e

A) a opção **-p** estiver presente ou o arquivo não tiver sido criado com atributo de gravação com sincronismo, a seguinte mensagem indica o erro:

Indice XX - erro
Esvazia arquivo (s/n)?

Se o usuário responder "S" ou "s", ou ainda se a opção **-y** for uma das opções de entrada do utilitário, o arquivo é esvaziado. Neste caso, o arquivo continua a existir, porém, sem registros. Se o usuário responder "N" ou "n", ou ainda se a opção **-n** estiver presente, o utilitário termina com a mensagem:

```
** manutenção incompleta - arq. inconsistente
```

deixando o arquivo marcado como inconsistente.

B) Se o arquivo tiver sido criado com atributo de gravação com sincronismo, o utilitário tenta resolver a inconsistência à revelia do usuário. Caso não consiga resolvê-la, o mesmo procedimento do item A é feito.

Ainda nesta fase, consiste-se o número de registros realmente existentes no arquivo com o número de registros que o descritor do arquivo indica. Se não coincidem, o descritor do arquivo é atualizado. O tamanho do arquivo também é atualizado no descritor do arquivo, de modo a refletir seu octeto de maior ordem.

Se a manutenção do arquivo terminar normalmente, a seguinte mensagem é emitida:

```
Indice XX - tamanho: ww octetos
            No. de níveis: YY
            Chave externa/interna
```

2. DADO INDIRETO

** Fase 1 - verificando índices

Nesta fase, os índices fornecidos pelo usuário, ou todos os índices pertencentes ao arquivo, são verificados quanto à sua estrutura (apontadores). Se alguma inconsistência for encontrada em um dos índices e

A) a opção **-p** estiver presente, ou o arquivo não tiver sido criado com atributo de gravação com sincronismo, a seguinte mensagem indica o erro:

```
Indice XX - erro
Reconstrói índice (s/n)?
```

Se o usuário responder "S" ou "s", ou ainda se a opção **-y** estiver presente, o índice XX é marcado para posterior reconstrução (Fase 3). Se o usuário responder "N" ou "n", ou ainda se a opção **-n** estiver presente, a seguinte mensagem é enviada:

```
Elimina índice (s/n)?
```

Se o usuário responder "S" ou "s", o índice é marcado para posterior eliminação (Fase 3). Se o usuário responder "N" ou "n", o manut termina com a mensagem:

```
** manutenção incompleta - arq. inconsistente
```

deixando o arquivo marcado como inconsistente. A pergunta sobre reconstrução do índice não é feita caso o índice com erro seja correspondente a chave externa.

B) se o arquivo tiver sido criado com o atributo de gravação com sincronismo, o utilitário tenta resolver a inconsistência à revelia do usuário. Caso não consiga resolvê-la, o mesmo procedimento do item A é feito.

Ao final da manutenção de cada índice, a seguinte mensagem indica término normal:

```
Indice XX - tamanho: ww octetos
            No. de níveis : YY
            Chave duplicada(unívoca)/
                  externa(interna)
```

Após a verificação de todos os índices, inicia-se a Fase 2:

```
** Fase 2 - verificando registros de dados
```

onde é feita a comparação entre os registros de dados com os índices, verificando se todo registro de dados possui uma chave correspondente em cada índice. Se alguma chave não foi encontrada em um dos índices e

A) for constatado que existe uma inconsistência em uma página de dados, e esta for irrecuperável, o utilitário termina com a mensagem:

```
Paginas de dados inconsistente - no. pagina
```

Se a inconsistência for recuperável através da eliminação da página, o utilitário perguntará se quer que elimine a página de dados.

B) for constatado que existe uma inconsistência no índice, o utilitário adota o mesmo procedimento descrito anteriormente na Fase 1, item A.

C) não existe inconsistência nas páginas de dados e nem no índice, a chave é então inserida no índice.

Ainda nesta fase, consiste-se o número de registros realmente existentes com o número de registros que o descritor do arquivo indica. Se não coincidem, o descritor do arquivo é atualizado. O tamanho do arquivo também é atualizado no descritor do arquivo, de modo a refletir seu octeto de maior ordem.

Na última fase, as operações de reconstrução ou eliminação de índices que ficaram pendentes são executadas:

```
** Fase 3
Reconstruindo índice XX
```

e/ou

```
Eliminado índice YY
```

3. DADO INDIRETO SEM ÍNDICES DEFINIDOS

Neste caso, onde só existem registros de dados, a seguinte mensagem é enviada:

** Fase 1 - verificando registros de dados

sendo feita a verificação da consistência de todas as páginas pertencentes ao arquivo. Se uma inconsistência for encontrada em alguma das páginas, o utilitário enviará a mensagem:

Pagina de dados inconsistente - no. da página
Elimina página de dados (s/n)?

Se o usuário responder "S" ou "s", ou ainda se **-y** for uma das opções de entrada, a página inconsistente é eliminada do arquivo. Se o usuário responder "N" ou "n", ou ainda se a opção **-n** estiver presente, o comando termina com a mensagem:

** manutenção incompleta - arq. inconsistente

deixando o arquivo marcado como inconsistente.

Consiste-se o número de registros realmente existentes no arquivo com o número de registros que o descritor do arquivo indica. Se não coincidem, o descritor do arquivo é atualizado. Também é atualizado no descritor do arquivo, o seu tamanho em octetos, que reflete o octeto de maior ordem.

CÓDIGOS DE RETORNO

0 - Arquivo consistente.

1 - Usuário respondeu "não" a uma pergunta e manut foi interompido.

2 - Manutenção interrompida (sinal).

3 - Organização do arquivo não é indexada.

4 - Índices inválidos.

5 - Diretório de páginas do descritor do arquivo inconsistente.

6 - Diretório de páginas do descritor do índice inconsistente.

7 - Dispositivo do arquivo não encontrado.

8 - Página de dados inconsistente.

9 - Não conseguiu reconstruir arquivo.

10- Usuário não tem permissão de gravação no arquivo.

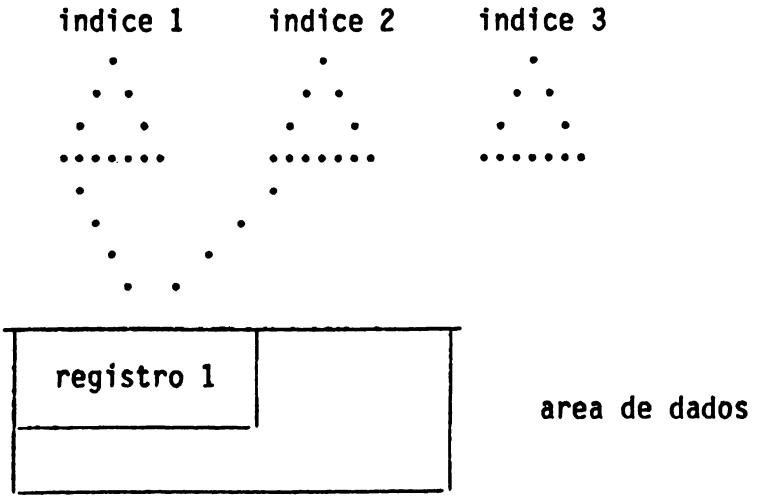
Maior que 100 - Término com erro.

OBSERVAÇÕES

. Se o comando precisar de resposta do usuário e a entrada-padrão não é um terminal, o comando assume a resposta "não".

. Se a opção **-i** estiver presente como parâmetro de entrada do comando, ou seja, nem todos os índices devem ser verificados/consistidos, é de inteira responsabilidade do usuário a consistência total do arquivo. Esta inconsistência pode ocorrer quando nem todos os índices do arquivo contêm uma chave para determinado registro de dados. O **manut** garante a consistência do arquivo para aqueles índices que foram verificados/consistidos. Aconselha-se que o usuário só faça uso desta opção, quando a operação seguinte ao **manut** for a de consulta pelos índices verificados.

Exemplo:



. Os índices 1 e 2 possuem chaves para o registro 1, mas o índice 3 não possui. O arquivo não está consistente como um todo.

. O comando **manut** não faz parte do pacote básico de comandos distribuído pela COBRA. É comercializado à parte, junto com o Sistema de Arquivamento Estendido SOX (SAX).

ARQUIVOS UTILIZADOS

Para recuperação de determinados erros, é criado o arquivo temporário "tempmanut" no mesmo diretório do arquivo a ser verificado / consistido. Este arquivo é removido ao final do utilitário independente do sucesso ou não da manutenção.

EXEMPLOS

```
. manut -y arqdir              (dado direto)


   ** Fase 1 - verificando índice

      Indice 1 - tamanho: xx octetos
                 no. de níveis: 3
                 chave unívoca/interna

   ** manutenção executada - arquivo consistente


. manut -n arqind              (dado indireto)


   ** Fase 1 - verificando índices

      Indice 1 - tamanho: xx octetos
                 no. níveis: 2
                 chave unívoca/interna
```

```
      Indice 2 - tamanho: xx octetos
                 no. níveis: 3
                 chave duplicada/interna

   ** Fase 2 - verificando registros de dados

   ** manutenção executada - arquivo consistente

. manut -n arqdados
          (dado indireto sem índices definidos)


   ** Fase 1 - verificando registros de dados

      Página de dados inconsistente - 3
      Elimina página de dados (s/n)? nao

 ** manutenção incompleta - arquivo inconsistente


. manut -pyi  arqind  1,3   (dado indireto)


   ** Fase 1 - verificando índices

      Indice 1 - tamanho: xx octetos
                 no. de níveis: 2
                 chave unívoca/externa

      Indice 3 - erro
      Reconstrói índice (s/n)? sim
```

```
** Fase 2 - verificando registros de dados

** Fase 3
   Reconstruindo índice 3

** manutenção executada - arquivo consistente
```

Página intencionalmente em branco.

MESG **Comando - Avançado**

mesg - habilita ou não o recebimento de mensagens.

SINTAXE

```
mesg [y|n]
```

DESCRIÇÃO

O comando **mesg** torna o terminal que executou o comando apto ou inapto a receber mensagens de outros usuários do sistema (por exemplo, através do comando **write**). Se utilizado sem argumento, informa o estado atual do terminal, sem alterá-lo.

OPÇÕES

Não existem.

ARGUMENTOS

y

Habilita o terminal a receber mensagens.

n

Desabilita o terminal a receber mensagens.

DIÁLOGO

Não existe.

CÓDIGOS DE RETORNO

0 - Apto a receber mensagens.

1 - Inapto a receber mensagens.

2 - Término provocado por erro.

OBSERVAÇÃO

O estado inicial de um terminal permite o recebimento de mensagens.

ARQUIVOS UTILIZADOS

/dev/tty*

VEJA TAMBÉM

write (comando)

MKDIR **Comando - Básico**

mkdir - cria diretórios.

SINTAXE

```
mkdir dir ...
```

DESCRIÇÃO

O comando **mkdir** cria os diretórios especificados. O diretório onde se deseje criar um subdiretório deve ter permissão de escrita para o comando. As entradas "." e ".." são automaticamente criadas nos diretórios.

OPÇÕES

Não existem.

ARGUMENTOS

dir ...

Nomes dos diretórios a serem criados.

DIÁLOGO

Não existe.

CÓDIGOS DE RETORNO

zero	- Término sem erro.
1 a 100	- Número de diretórios não criados (Módulo 101).
maior que 100	- Término com erro.

ARQUIVOS UTILIZADOS

Não existem.

VEJA TAMBÉM

rm(comando)

MKFS **Comando - Administração**

mkfs - cria sistema de arquivos em um volume.

SINTAXE

```
/etc/mkfs especial blocos [:bcas]

/etc/mkfs especial proto
```

DESCRIÇÃO

O comando **mkfs** cria um sistema de arquivos no volume presente no dispositivo **especial**, de acordo com as especificações dadas no restante da linha de comando.

Se o segundo argumento começa com uma cadeia de dígitos (formato 1), **mkfs** cria o sistema de arquivos com tamanho de **blocos** blocos (de 512 octetos) e **BCAs**, Bloco de Controle de Arquivo, contendo apenas o diretório raiz vazio. O programa de carga do sistema não é copiado.

Se o segundo argumento é um nome de arquivo que possa ser aberto (formato 2), **mkfs** assume que o mesmo é um arquivo protótipo **proto**, e obterá do mesmo as especificações do sistema de arquivos a ser criado.

O arquivo protótipo contém símbolos separados por brancos ou carateres de avanço de linha (**LF**). O primeiro símbolo é o nome de um arquivo para ser copiado a partir do bloco zero, como o programa de carga do sistema ("bootstrap"). O segundo símbolo é um número que especifica o tamanho do sistema de arquivos a ser criado, em blocos de 512 octetos. O terceiro símbolo é o número de Blocos de Controle de Arquivo (BCA) do sistema de arquivos. O próximo conjunto de símbolos compreende a especificação do arquivo raiz. Especificações de arquivos consistem em símbolos dando o modo do arquivo, identificador de usuário, identificador de grupo e o conteúdo inicial do arquivo. A sintaxe do campo de conteúdo depende do modo do arquivo.

O símbolo de modo de um arquivo é uma cadeia de seis caracteres. O primeiro caractere especifica o tipo do arquivo:

b : bloco especial
c : caractere especial
d : diretório
- : arquivo regular "stream"

O segundo caractere do modo é **u** ou **-** para especificar ou não o modo de alteração do identificador de usuário em execução. O terceiro caractere é **g** ou **-** (hífen) para especificar ou não o modo de alteração do identificador de grupo em execução. O resto do modo é um número octal de 3 dígitos com as permissões de leitura, escrita e execução (busca) para o dono, grupo e outros.

Dois números decimais vêm após o modo; eles especificam os identificadores de usuário e de grupo do arquivo.

Se o arquivo é um arquivo-regular, o próximo símbolo é um nome de caminho de onde é copiado o conteúdo do arquivo. Se o arquivo é um arquivo-especial tipo bloco ou caractere, seguem-se dois números decimais que especificam o tipo e o subtipo do dispositivo associado. Se o arquivo é um diretório, **mkfs** constrói as entradas e lê uma lista de nomes e (recursivamente) de especificações de arquivo para as entradas no diretório. A busca termina ao ser lido o símbolo **$**.

OPÇÕES

Não existem.

ARGUMENTOS

especial

Nome do arquivo caractere especial associado ao dispositivo onde será criado o sistema de arquivos.

blocos

Número de blocos (de 512 octetos) que serão utilizados para o sistema de arquivos. Normalmente é o número total de blocos do dispositivo.

bcas

Número de BCAs que serão reservados para o sistema de arquivos. Se não especificado, assume o valor default 100.

proto

Arquivo que contém as caraterísticas desejadas para o sistema de arquivos.

DIÁLOGO

Não existe.

CÓDIGOS DE RETORNO

zero - término sem erro.

diferente de zero - término com erro.

OBSERVAÇÕES

. Se o dispositivo não for um disco winchester, antes de ser criado o sistema de arquivos, todos os blocos do volume são testados, através de leituras e gravações. Os blocos com erro são marcados no mapa de blocos como ocupados.

. A geração de um **boot strap** não vazio em disco rígido, não é possível no SOX-PC e SOX-500.

ARQUIVOS UTILIZADOS

Não existem.

EXEMPLO

Exemplo de um arquivo de protótipo:

```
/etc/boot
360 100
d--755  0  0
usr       d--755  2  2
          sh      ---755  2  2  /bin/sh
          util    d--755  8  4
                  $
          b0      b--644  2  2  1  2
          c0      c--644  2  2  17 2
          $
$
```

Página intencionalmente em branco.

MKNOD **Comando - Administração**

mknod - cria arquivos especiais.

SINTAXE

/etc/mknod nome b | c tipo subtipo

/etc/mknod nome p

DESCRIÇÃO

O comando **mknod** cria arquivos especiais. A primeira forma é usada para criar arquivos especiais tipo caractere (argumento **c**) ou arquivos especiais tipo bloco (argumento **b**). Os dois ultímos argumentos são números que especificam o tipo e o subtipo (unidade, número da linha, etc.) do dispositivo desejado. Os números podem ser representados tanto na forma decimal quanto octal (número precedido por zero).

A segunda forma sintática é usada para criar arquivos especiais tipo fifo.

OPÇÕES

Não existem.

ARGUMENTOS

nome

Nome do arquivo especial a ser criado.

b, c, p

Indicam arquivo especial tipo bloco, caractere e FIFO respectivamente.

tipo

Número usado para especificar o tipo de um dispositivo (disco flexível, impressora, etc...).

subtipo

Número usado para especificar um dispositivo numa família de dispositivos (número de linha, número da unidade de disco flexível, etc.).

DIÁLOGO

Não existe.

CÓDIGOS DE RETORNO

zero - término sem erro.

diferente de zero - término com erro.

OBSERVAÇÃO

A primeira forma do comando **mknod** (criação do arquivo especial) só pode ser executada por superusuário.

ARQUIVOS UTILIZADOS

Não existem.

VEJA TAMBÉM

Manual Administrador do Sistema SOX (SX03)

Página intencionalmente em branco.

MOUNT **Comando - Administração**

mount - monta sistemas de arquivos.

SINTAXE

```
/etc/mount [especial dir [-r]]
```

DESCRIÇÃO

O comando **mount** é utilizado para montar um volume residente no arquivo **especial**. O diretório já deve existir, e torna-se o diretório-raiz do novo volume montado.

Sempre que uma montagem é completada com sucesso, **mount** atualiza a tabela de volumes montados **/etc/mnttab** com informações a respeito da mesma, quais sejam: o dispositivo utilizado, o diretório sobre o qual o volume foi montado, a disponibilidade de acesso ao volume (leitura/escrita ou leitura somente) e data da montagem.

O comando **mount** sem argumentos lista a(s) entrada(s) da tabela **/etc/mnttab.**

OPÇÕES

-r

Opção de montagem do volume somente para leitura. Volumes fisicamente protegidos contra escrita e volumes de fita magnética devem ser montados dessa forma, caso contrário erros irão ocorrer quando tempos de acesso forem atualizados, havendo ou não tentativas explícitas de escrita no volume.

ARGUMENTOS

especial

Nome de um arquivo especial tipo bloco , que deve estar associado a um dispositivo existente.

dir

Nome de um arquivo tipo diretório, que não pode ser o diretório corrente do usuário, nem ser raiz de um outro volume já montado.

DIÁLOGO

Tecle <**RET**>quando o volume estiver montado na unidade.

CÓDIGOS DE RETORNO

zero - término sem erro.

diferente de zero - término com erro.

OBSERVAÇÃO

O comando emite mensagem de erro se o volume a ser montado já estiver montado sob outro diretório.

ARQUIVOS UTILIZADOS

/etc/mnttab

Página intencionalmente em branco.

MV **Comando - Básico**

mv - move arquivos.

SINTAXE

```
mv [-fl] arq1 [arq2 ...] destino
```

DESCRIÇÃO

O comando **mv** move ou renomeia os arquivos especificados.

Se **destino** não é um diretório, apenas **arq1** pode ser especificado, e, neste caso, **arq1** é renomeado para **destino**. Se **arq1** é um diretório, sua renomeação só é possível se **destino** não existir e se ambos pertencerem ao mesmo diretório-pai.

Se **destino** é um diretório, todos os arquivos especificados são movidos para este diretório e apagados do diretório original.

Se **destino** é um arquivo e o seu modo não permite escrita, o modo é impresso e é feita uma pergunta ao usuário. Caso a resposta comece por um dos caracteres do conjunto "sSyY"(sim) e o usuário tem permissão de escrita em **destino**, a movimentação é realizada. Nenhuma pergunta é feita e a movimentação é efetuada (quando permitida), se a opção **-f** é usada, ou se a entrada padrão não está associada a terminal.

OPÇÕES

-f

Suprime a pergunta e assume que a resposta é sim.

-l

Mostra cada arquivo movido.

ARGUMENTOS

arq1 arq2...

Diretórios ou arquivos a serem movidos.

destino

Arquivo ou diretório (novo nome ou destino da movimentação).

DIÁLOGO

arquivo com modo modo **:continua ?**

CÓDIGOS DE RETORNO

zero - término sem erro.

diferente de zero - término com erro.

OBSERVAÇÕES

. Se **arq1** for um arquivo e **destino** uma associação para outro arquivo com associações, as outras associações permanecem, e **destino** torna-se um novo arquivo.

. Se **arq1** e **destino** residem em volumes diferentes, **mv** copia o arquivo e deleta o original. Neste caso, são perdidas quaisquer associações que o mesmo possuía.

ARQUIVOS UTILIZADOS

Não existem.

VEJA TAMBÉM

chmod (comando)
cp (comando)
cpio (comando)
ln (comando)
mvdir (comando)
rm (comando)

Página intencionalmente em branco.

MVDIR **Comando - Administração**

mvdir - move um diretório.

SINTAXE

```
/etc/mvdir dir  nome
```

DESCRIÇÃO

O comando **mvdir** move diretórios dentro do mesmo volume.

O argumento dir tem que ser um diretório e o argumento nome não pode ser um arquivo que já exista. Se nome é um diretório, então dir é movido para nome/dir, desde que não exista tal arquivo ou diretório.

Se nome não existe, ele é criado como diretório e dir é movido para nome (como se dir fosse renomeado).

OPÇÕES

Não existem.

ARGUMENTOS

dir

Nome do diretório-origem.

nome

Nome do diretório-destino.

DIÁLOGO

Não existe.

CÓDIGOS DE RETORNO

zero - término sem erro.

diferente de zero - término com erro.

OBSERVAÇÕES

. Somente o superusuário pode executar este comando.

. O nome ou diretório não podem ser subconjuntos um do outro; por exemplo, /x/y não pode ser movido para /x/y/z, ou vice-versa.

ARQUIVOS UTILIZADOS

Não existem.

NCHECK **Comando - Administração**

ncheck - lista os nomes dos arquivos associados aos números dos BCAs.

SINTAXE

```
ncheck [-i n_bca] [-a] [-s] [sist_arq]
```

DESCRIÇÃO

O comando **ncheck** lista os nomes dos arquivos associados aos números dos BCAs.

Este comando sem argumentos gera uma lista de caminho de arquivos associado ao seu número do BCA, para todos os arquivos do sistema de arquivos default (**/dev/dw0**).

OPÇÕES

-i

Fornece o relatório com os nomes dos arquivos cujos números dos BCAs são fornecidos.

-a

Permite que sejam listados os nomes dos arquivos "." e ".." de cada diretório.

-s

Fornece o relatório com os nomes dos arquivos especiais e dos arquivos com modo **set-usr-id.**

ARGUMENTOS

n_bca

Indica o número do bca do arquivo a ser pesquisado. Podem ser especificados no máximo dez números de bca separados pelo caractere espaço.

sist_arq

Especifica o sistema de arquivos. Se omitido é assumido **/dev/dw0.**

DIÁLOGO

Não existe.

CÓDIGOS DE RETORNO

zero - término sem erro.

diferente de zero - término com erro.

OBSERVAÇÕES

. Os nomes dos arquivos diretórios são seguidos por "**/..**".

. O comando **ncheck** só pode ser executado para um sistema de arquivos que não tenha nenhum outro sistema de arquivos montado nele.

ARQUIVOS UTILIZADOS

Não existem.

VEJA TAMBEM

fsck (comando)

EXEMPLOS

. ncheck -a

Lista todos os nomes dos arquivos do sistema de arquivos default (**/dev/dw0**), inclusive os arquivos "." e ".." de cada diretório.

. ncheck -s /dev/df0

Fornece o relatório com os nomes dos arquivos especiais e dos arquivos com modo **set-usr-id** do sistema de arquivos **/dev/df0.**

Página intencionalmente em branco.

NEWS **Comando - Uso Geral**

news - exibe notícias.

SINTAXE

```
news [-a] [-n] [-s] [arq...]
```

DESCRIÇÃO

O comando **news** informa as notícias arquivadas no diretório de notícias do sistema (**/usr/news**).

Quando chamado sem argumentos, o comando **news** exibe o conteúdo de todos os arquivos existentes no diretório de notícias do sistema, cuja data de criação ou alteração seja mais recente do que a última vez que o usuário ativou o comando **news**. Os arquivos mais recentes são exibidos primeiro, sendo cada um precedido por um cabeçalho apropriado.

O comando **news** controla a data em que o usuário ativou o comando pela última vez, através da data de modificação do arquivo .news_time do diretório do usuário. Somente os arquivos de notícias mais recentes do que a data do arquivo .news_time são considerados notícias correntes.

OPÇÕES

-a

Imprime todos os arquivos de notícias existentes no diretório, inclusive os anteriores aos correntes, e não altera a data da última ativação do comando.

-n

Exibe apenas os nomes dos arquivos de notícias correntes sem imprimir seus conteúdos, e altera a data da última ativação do comando.

-s

Informa o número de arquivos de notícias correntes existentes no diretório de notícias, sem exibir seus nomes ou conteúdos, e não altera a data da última ativação do comando.

ARGUMENTO

arq

Arquivo de notícias.

DIÁLOGO

Não existe.

CÓDIGOS DE RETORNO

Não existem.

OBSERVAÇÃO

Se uma interrupção (DEL ou BREAK) é gerada durante a listagem, é exibida na tela a mensagem **stops** e a próxima notícia é iniciada. Outra interrupção seguida à primeira causa o término do programa.

ARQUIVOS UTILIZADOS

/etc/profile
$HOME/.news_time

EXEMPLOS

. **news -a**

Lista todos os arquivos de notícias existentes no diretório **/usr/news**, inclusive os arquivos anteriores aos correntes, e não altera a data da última ativação do comando.

. **news -n**

Lista os nomes dos arquivos de notícias correntes sem exibir seus conteúdos, e altera a data da última ativação do comando.

Página intencionalmente em branco.

NEWGRP **Comando - Avançado**

newgrp - muda para novo grupo.

SINTAXE

```
newgrp [-] [grupo]
```

DESCRIÇÃO

O comando **newgrp** troca a identificação de grupo do usuário. O usuário continua com sua sessão aberta e no mesmo diretório corrente, mas os cálculos das permissões de acesso aos arquivos são efetuados em relação aos novos identificadores de grupo, real e efetivo.

Variáveis exportadas do ambiente mantêm seus valores após a execução do comando **newgrp**. As variáveis não exportadas do ambiente são reiniciadas com seus valores default, ou recebem o valor nulo. As variáveis ambientais (tais como PS1, PS2, PATH, MAIL e HOME), se não forem exportadas, são iniciadas com seus valores default.

Sem argumentos o comando **newgrp** restaura o identificador de grupo do usuário para aquele especificado no arquivo /etc/passwd.

Se o primeiro argumento do comando **newgrp** é um -, o ambiente é trocado pelo ambiente que seria esperado caso o usuário tivesse novamente aberto sessão.

OPÇÕES

Não existem.

ARGUMENTOS

-

Reinicia o ambiente, como se o usuário tivesse aberto uma nova sessão.

grupo

Identificador de grupo do usuário.

DIÁLOGO

Não existe.

CÓDIGOS DE RETORNO

zero - término sem erro.

diferente de zero - término com erro.

ARQUIVOS UTILIZADOS

/etc/group
/etc/passwd

VEJA TAMBÉM

sh (comando)

NICE **Comando - Administração**

nice - executa comando com baixa prioridade.

SINTAXE

```
nice [-incremento] comando
```

DESCRIÇÃO

O comando **nice** executa o argumento comando, diminuindo sua prioridade de despacho..

OPÇÕES

Não existem.

ARGUMENTOS

-incremento

O incremento é um inteiro positivo menor que NZERO. Na sua falta, o valor utilizado é metade de NZERO (arredondado para cima). Um incremento maior que (NZERO-1) é equivalente a (NZERO-1).

comando

Comando a ser executado.

DIÁLOGO

Não existe.

CÓDIGOS DE RETORNO

Os mesmos do comando executado.

OBSERVAÇÕES

. Superusuários podem executar comandos com prioridade mais alta que a normal, usando incremento negativo, por exemplo:
nice --4 cmd.

. O valor do parâmetro NZERO da instalação encontra-se em /usr/include/limits.h.

ARQUIVOS UTILIZADOS

Não existem.

VEJA TAMBÉM

nice (chamada)

EXEMPLOS

Compilando com baixa prioridade:

nice cc -o teste teste.c

NL **Comando - Básico**

nl - filtro de numeração de linhas.

SINTAXE

```
nl  [-btipo] [-btipo] [-ftipo] [-p]
    [-vinic] [-iincr] [-wlarg] [-nformat]
    [-ssep] [-lnum] [-ddelim] [arq]
```

DESCRIÇÃO

O comando **nl** lê linhas do arquivo especificado ou da entrada-padrão, se nenhum arquivo foi especificado, e numera linha a linha deste arquivo na saída-padrão. A numeração é feita à esquerda da linha, de acordo com as opções em efeito.

nl considera o texto que lê em termos de páginas lógicas. A numeração é reiniciada no começo de cada página lógica. Uma página lógica consiste em seções de cabeçalho, corpo e rodapé, permitindo-se seções vazias. Diferentes opções de numeração de linha estão disponíveis de forma independente para cabeçalho, corpo e rodapé (por exemplo, não numerar cabeçalho e rodapé, enquanto numera linhas em branco apenas do corpo).

O começo das seções de uma página lógica é indicado por linhas de entrada contendo apenas o(s) seguinte(s) caractere(s) delimitador(es):

Linha	Início de
\:\:\:	cabeçalho
\:\:	corpo
\:	rodapé

A menos que especificado em contrário, **nl** assume que o texto sendo lido consiste em um único corpo de página lógica.

OPÇÕES

As opções podem aparecer em qualquer ordem e podem ser misturadas com um nome de arquivo.

-btipo

Definição do controle de numeração numa seção de corpo. São os seguintes os valores de **tipo**:

a - numera todas as linhas.

t - numera só as linhas de texto imprimível (default).

n - não numera nenhuma linha.

p cadeia - só numera as linhas que contenham a expressão regular especificada em **cadeia.**

-htipo

Igual a **-btipo** aplicado em cabeçalhos, sendo default **n.**

-ftipo

Igual a **-btipo** aplicado em rodapés, sendo default **n.**

-ddelim

Muda o delimitador default de início de página lógica para **delim.**

delim é composto por dois caracteres e não pode ser separado de **-d** por branco. Se só for dado um caractere para especificar **delim**, o segundo permanece o default (:). Para fornecer uma contra-barra, entre com duas contra-barras.

-p

Não ocasiona um reinício de numeração em um delimitador de página lógica.

-vinic

inic informa qual o número inicial de numeração de cada página lógica. O default é 1.

-iincr

incr informa qual o incremento de numeração para cada página lógica. O default é 1.

-wlarg

larg especifica o número de dígitos para a numeração da linha. O default é 6.

-nformat

Formato da numeração dentro da seção. Os valores de **format** são:

ln - Justificado à esquerda, suprimindo zeros à esquerda.

rn - Justificado à direita, suprimindo zeros à esquerda (default).

rz - Justificado à direita, mantendo os zeros à esquerda.

-ssep

sep define o(s) caractere(s) de separação entre o campo reservado para a numeração e o texto da linha. O default é **HT** (tabulação horizontal).

-lnum

num é o número de linhas em branco a serem contadas como uma para fins de numeração. O default é 1.

ARGUMENTO

arq

Arquivo a numerar. Apenas um arquivo pode ser especificado.

DIÁLOGO

Não existe.

CÓDIGOS DE RETORNO

zero - término sem erro.

diferente de zero - término com erro.

ARQUIVOS UTILIZADOS

Não existem.

EXEMPLO

nl -l2 -ha -ba arq

Faz com que somente a segunda linha em branco adjacente seja numerada.

Página intencionalmente em branco.

NM **Comando - Desenvolvimento**

nm - imprime a tabela de símbolos de arquivo-objeto.

SINTAXE

```
nm [-efouxV] arq...
```

DESCRIÇÃO

O comando **nm** lista na saída padrão a tabela de símbolos de arquivo-objeto (absolutos ou relocáveis) e/ou bibliotecas (de arquivos-objeto absolutos ou relocáveis) criadas pelo editor de bibliotecas **ar**.

Se o argumento passado ao utilitário for uma biblioteca, serão listadas as tabelas de símbolos de todos os arquivos-objeto membros da biblioteca.

Para cada símbolo são listadas, pelo menos, as seguintes informações:

nome : nome do símbolo.

valor : valor expresso como deslocamento ou endereço dentro da seção que define o símbolo.

classe : classe de armazenamento de símbolo.

tamanho : tamanho em octetos, se disponível.

tipo : tipo do símbolo conforme definido no fonte.

linha : linha de definição do símbolo no fonte.

seção : seção de definição do símbolo.

OPÇÕES

Sem opções, o comando **nm** lista os valores numéricos em decimal.

-o

Imprime valor e tamanho do símbolo em octal.

-e

Imprime somente os símbolos externos e estáticos.

-f

Imprime as entradas **.text**, **.data** e **.bss**, que normalmente são suprimidas.

-u

Imprime somente os símbolos indefinidos.

-x

Imprime o valor e tamanho do símbolo em hexadecimal.

-V

Imprime a versão do comando nm que está sendo executado, na saída-padrão de erros.

ARGUMENTO

arq...

Arquivos-objeto e/ou bibliotecas a serem processados.

DIÁLOGO

Não existe.

CÓDIGOS DE RETORNO

zero	- término sem erro.
diferente de zero	- término com erro.

ARQUIVOS UTILIZADOS

Não existem.

VEJA TAMBÉM

ar (comando)
cc (comando)
ld (comando)

Página intencionalmente em branco.

NOHUP **Comando - Básico**

nohup - executa um comando sem interrupção externa.

SINTAXE

```
nohup comando [arg...]
```

DESCRIÇÃO

O comando **nohup** executa **comando**, ignorando os sinais SIGHUP e SIGQUIT. Se a saída não for redirecionada pelo usuário, a saída-padrão e a saída-padrão de erros serão enviadas para o arquivo **nohup.out**. Se no diretório-corrente não for permitida a escrita em **nohup.out**, a saída-padrão será redirecionada para **$HOME/nohup.out**.

Se a saída-padrão estava associada a terminal, o comando grava na saída-padrão de erros uma mensagem de advertência, avisando que a saída do comando usado com **nohup** foi enviada para **nohup.out** ou **$HOME/nohup.out**. Esta mensagem é escrita antes das saídas serem redirecionadas por **nohup**.

OPÇÕES

Não existem.

ARGUMENTO

comando [arg]

Qualquer comando com ou sem argumentos, ou arquivos de comando **shell.**

DIÁLOGO

Não existe.

CÓDIGOS DE RETORNO

zero - término sem erro.

diferente de zero - término com erro.

OBSERVAÇÃO

Para se aplicar **nohup** a canalizações ou listas de comandos, deve-se colocar as canalizações e listas de comandos em um arquivo de comandos. Assim, este arquivo pode ser executado como **comando**, e **nohup** se aplica a todo o conteúdo do arquivo.

ARQUIVOS UTILIZADOS

nohup.out
$HOME/nohup.out

VEJA TAMBÉM

sh (comando)
signal (chamada)

Página intencionalmente em branco.

OD **Comando - Avançado**

od - "dump" em octal.

SINTAXE

```
od [-bcdosx] [arq] [[+]desloc[.][b]]
```

DESCRIÇÃO

O comando **od** realiza um "dump" do arquivo **arq** em um ou mais formatos. Se este argumento não for especificado, será usada a entrada-padrão. A opção default é **-o.**

OPÇÕES

-b

Interpreta octetos em octal.

-c

Interpreta octetos em ASCII. Alguns caracteres não-gráficos serão exibidos segundo o padrão da linguagem C:

\0 - nulo (ASCII NUL)

\b - volta caractere (ASCII BS)

\f - alimentação de formulário (ASCII FF)

\n - avanço de linha (ASCII LF)

\r - retorno de carro (ASCII CR)

\t - tabulação horizontal (ASCII HT)

Os outros caracteres não-gráficos aparecerão como um número octal de três dígitos.

-d

Interpreta palavras em decimal sem sinal.

-o

Interpreta palavras em octal.

-s

Interpreta palavras em decimal com sinal.

-x

Interpreta palavras em hexadecimal.

ARGUMENTOS

arq

Arquivo do qual se deseja o "dump".

desloc

O argumento **desloc** especifica o deslocamento, a partir do início do arquivo, de começo do "dump". Normalmente este argumento é interpretado como octeto em octal. Se for usado com **.**, este número será considerado decimal. Se usado com **b**, o deslocamento será interpretado como em unidades de 512 octetos. Se o argumento **arq** for omitido, o argumento **desloc** tem que ser precedido por **+**.

A base do argumento **desloc** (octal ou decimal) determina a base em que será impresso o deslocamento de cada linha do "dump".

DIÁLOGO

Não existe.

CÓDIGOS DE RETORNO

zero - término sem erro.

diferente de zero - término com erro.

OBSERVAÇÃO

No contexto das opções, "palavra" representa uma unidade de 16 bits, independente do tamanho de palavra da máquina.

ARQUIVOS UTILIZADOS

Não existem.

EXEMPLOS

. od -xc arquivo 10.

Gera o "dump" do arquivo **arquivo** em hexadecimal e ASCII a partir do décimo octeto.

. od -cx +10. < arquivo

Gera o "dump" do arquivo **arquivo** em hexadecimal e ASCII, a partir do décimo octeto.

PACK **Comando - Básico**

pack - compacta arquivos.

SINTAXE

```
pack [-] [-f] arq...
```

DESCRIÇÃO

O comando **pack** cria a partir do arquivo de entrada arq um arquivo compactado de nome arq.z, com a mesma permissão, datas de criação e modificação e dono de arq. Após a compactação com sucesso arq será removido do diretório.

Arquivos compactados, podem ser descompactados através dos comandos **unpack** ou **pcat.**

Não é permitida a compactação de arquivos para os quais foram estabelecidas associações.

OPÇÕES

O comando **pack** passado sem opções compacta o conteúdo de um ou mais arquivos passados como argumentos.

-

Seta e "resetar" um flag interno, para determinar o número de vezes que cada octeto é usado, sua freqüência relativa e o código para que este seja impresso na saída-padrão.

-f

Força a compactação de um arquivo passado como argumento, quando alguma característica deste, não resulta em economia de espaço. É útil quando se deseja compactar todos os arquivos de um diretório.

ARGUMENTOS

arq

Arquivo que se deseja compactar.

DIÁLOGO

Não existe.

CÓDIGOS DE RETORNO

O comando **pack** retorna o número de arquivos que não foram compactados.

OBSERVAÇÕES

. O nível de compactação obtido depende do tamanho do arquivo de entrada e da freqüência de distribuição dos caracteres.

. Geralmente arquivos de texto são reduzidos de 60% a 75% de seu tamanho original. Módulos de carga, que usam um conjunto maior de caracteres e têm uma distribuição mais uniforme de caracteres, apresentam baixa compactação. As versões compactadas têm cerca de 90% de seu tamanho original.

. O último segmento do nome do arquivo não deve conter mais que (NOME_MAX)-2, ou seja 12, caracteres de forma a permitir que seja acrescentado o sufixo ".z".

ARQUIVOS UTILIZADOS

Não existem.

VEJA TAMBÉM

cat (comando)

EXEMPLOS

. pack arq

Compacta arq.

. pack -f /lixo/arq

Força a compactação de arq passado como argumento.

. pack - arq

Compacta o arq passado como argumento, exibindo na saída-padrão informações adicionais sobre a compactação.

. pack -f /usr/carlos/*

Força a compactação de todos os arquivos do diretório /usr/carlos.

PASSWD **Comando - Avançado**

passwd - troca a senha da sessão.

SINTAXE

```
passwd  [usuário]
```

DESCRIÇÃO

O comando **passwd** troca ou instala a senha associada com o nome do usuário dado. Usuários comuns só podem trocar sua própria senha.

O comando pede a usuários comuns a senha antiga, se existir. Em seguida, pede a nova senha, que deve ser digitada duas vezes, para se evitar erros. Logo após a primeira digitação, o comando verifica se a nova senha está sintaticamente correta.

Quando a nova senha for lida pela segunda vez, as duas cópias serão comparadas. Se não forem idênticas, o ciclo de solicitação da nova senha é repetido pelo menos mais duas vezes.

OPÇÕES

Não existem.

ARGUMENTO

usuário

Usuário de que se quer alterar a senha. Se omitido, assume o usuário que executou o comando.

DIÁLOGO

Antiga senha:

Nova senha:

Confirme nova senha:

CÓDIGOS DE RETORNO

zero - término sem erro.

diferente de zero - término com erro.

OBSERVAÇÕES

. A senha deve ter pelo menos quatro caracteres. Se maior, somente os oito primeiros serão considerados.

. Se o usuário for privilegiado, a senha antiga não será pedida, e a sintaxe da nova não será verificada.

. Uma senha nula é interpretada como usuário sem senha. Somente o superusuário pode retirar uma senha, bastando para isso digitar um **LF** em resposta aos pedidos de nova senha e de confirmação.

ARQUIVO UTILIZADO

/etc/passwd

Página intencionalmente em branco.

PASTE — Comando - Básico

paste - combina linhas correspondentes de vários arquivos ou linhas subseqüentes de um arquivo.

SINTAXE

```
paste  arq1  arq2...

paste  -dlista  arq1  arq2...

paste  -s  [-dlista]  arq...
```

DESCRIÇÃO

O comando **paste** nos dois primeiros formatos combina as linhas correspondentes dos arquivos de entrada. No terceiro são combinadas as linhas subseqüentes do mesmo arquivo.

Caso a saída do comando seja a saída-padrão, pode haver um redirecionamento ou mesmo ser usada uma canalização.

Se o caractere - (hífen) for passado como argumento (arq) é adotada a entrada-padrão.

O comando **paste** fornece como saída uma tabela formada por colunas separadas pelo caractere **TAB** (default) ou pelo(s) especificado(s) na opção **-d**.

OPÇÕES

-d

Permite especificar os caracteres delimitadores das colunas. Se omitida, **TAB** é assumido como caractere delimitador. Nos dois primeiros formatos todos os caracteres nova linha são substituídos por **TAB**, exceto os do último arquivo. No terceiro formato, apenas o último caractere nova linha de cada arquivo não é alterado.

-s

Promove a concatenação de linhas subseqüentes. Se não tiver sido especificada a lista de caracteres substitutiva em -d, assume-se o caracter de tabulação.

ARGUMENTOS

lista

São caracteres que, sucedendo a opção **-d**, será usada circularmente substituindo o caractere TAB como delimitador. Caracteres especiais da shell devem estar entre aspas.

arq

Arquivo a ser combinado.

DIÁLOGO

Não existe.

CÓDIGOS DE RETORNO

zero - término sem erro.

diferente de zero - término com erro.

ARQUIVOS UTILIZADOS

Não existem.

VEJA TAMBÉM

cut (comando)
grep (comando)
pr (comando)

EXEMPLO

Para dois arquivos arq1 e arq2 cujos os conteúdos são:

arq1	arq2
aaaaaa	11111
bbbbbb	22222
cccccc	33333
dddddd	44444
	55555

usando os comandos a seguir, obtemos:

. paste -dAB arq2 arq1 arq2

```
11111AaaaaaB11111
22222AbbbbbB22222
33333AcccccB33333
44444AdddddB44444
55555AB55555
```

PCAT **Comando - Básico**

pcat - lista arquivos compactados.

SINTAXE

```
pcat arq...
```

DESCRIÇÃO

O comando **pcat** lista o conteúdo dos arquivos compactados anteriormente pelo comando pack, não podendo ser usado como filtro.

Os arquivos são descompactados e listados na saída-padrão.

OPÇÕES

Não existem.

ARGUMENTO

arq

Arquivo compactado que se deseja listar. Este pode ser passado da forma arq.z ou somente arq.

DIÁLOGO

Não existe.

CÓDIGOS DE RETORNO

O comando **pcat** retorna o número de arquivos que não puderam ser descompactados.

ARQUIVOS UTILIZADOS

Não existem.

VEJA TAMBÉM

cat (comando)

EXEMPLOS

. pcat arq1.z

Lista o conteúdo de arq1.z.

. pcat arq1

Lista o conteúdo de arq1.z.

. pcat arq1 arq2 arq3 arq4

Lista o conteúdo de arq1.z, arq2.z, arq3.z e arq4.z.

. pcat arq1.z>teste

Cria o arquivo "teste" não compactado sem remover o arquivo arq1.z.

. pcat arq1>teste

Cria o arquivo "teste" não compactado sem remover o arquivo arq1.z.

Página intencionalmente em branco.

PG **Comando - Básico**

pg - imprime arquivos.

SINTAXE

```
pg [-cefns] [+lin] [-tam] [-prad] [+/exp/]
    [arq]
```

DESCRIÇÃO

O comando **pg** é um filtro que permite examinar arquivos na tela de um terminal. Se o nome do arquivo for omitido ou fornecido "-" (hífen), os nomes dos arquivos serão lidos da entrada-padrão, um em cada linha até que o fim de arquivo seja encontrado.

Após a apresentação de cada tela, **pg** envia um "pronto" especial, esperando uma resposta do usuário que definirá a ação seguinte. Se o usuário, digitar um **carriage return**, será mostrada outra página.

Este comando permite, também, voltar páginas. Para saber os atributos do terminal, **pg** pesquisa o arquivo TERMINFO de descrição do terminal, partir da variável de ambiente TERM. Se TERM não for definida, ele assume **dumb**.

OPÇÕES

O comando **pg** sem opções mostra, no vídeo, o conteúdo do arquivo, dividido em janelas e a cada fim de página espera um **carriage return** para prosseguir.

As opções, a seguir, podem ser usadas na sintaxe do comando.

-c

Coloca o cursor no canto superior esquerdo (home) e limpa a tela antes da impressão de cada página. Esta opção não é válida se a característica clear_screen não foi especificado para este terminal no arquivo TERMINFO correspondente.

-e

Elimina a pausa após a impressão de cada arquivo.

-f

A opção **-f** impede o particionamento das linhas. Normalmente, **pg** divide linhas que sejam maiores que o tamanho da tela, porém, algumas seqüências de caracteres no texto sendo impresso, podem gerar resultados indesejáveis.

-n

Os comandos geralmente são seguidos pelo caractere nova linha. Esta opção permite um fim automático, tão logo uma letra, correspondente ao comando interno seja digitada.

-s

Imprime todas as mensagens e os "prontos" em vídeo no reverso.

+

Permite especificar a linha inicial do texto.

-

Especifica o tamanho da janela em número de linhas. Se omitido é assumido 23 linhas.

-p

Substitui o "pronto" especial, cujo default é : (dois pontos), para aquele definido por "cad".

ARGUMENTOS

lin

Inteiro que especifica a linha a partir da qual é iniciada a impressão (inclusive).

tam

Inteiro maior que zero que especifica o tamanho da janela em número linhas.

cad

Cadeia que substitui o "pronto". O número da página corrente constará do sinal de pronto de **pg** caso "**%d**" esteja presente na cadeia. Esta substituicao é realizada somente na primeira ocorrência de"**%d**".

exp

Inicia a impressão a partir da primeira linha que contenha a expressão-regular "exp".

arq

Cadeia de caracteres que especifica o arquivo que será impresso. Se for omitido o nome do arquivo ou for usado o caractere -, a entrada-padrão será utilizada.

COMANDOS INTERNOS

Após a impressão de cada página, o comando faz uma pausa quando o usuário poderá digitar os comandos internos ao **pg**. Estes comandos podem ser divididos em três categorias:

. Análise anterior;

. Procura de expressões;

. Modifica o ambiente.

ANÁLISE ANTERIOR

Neste caso o comando toma o endereço, opcionalmente um número com sinal, indicando a partir de onde o texto deve ser impresso. O endereço pode ser interpretado como páginas ou linhas, dependendo do comando. Um endereço com sinal indica um ponto relativo à página ou linha corrente. Endereços sem sinal indicam endereços relativos ao início do arquivo. Todos os comandos possuem endereços default.

Os comandos são definidos a seguir:

+num{<newline>, espaço}

Imprime uma página. O endereço "num" especifica a página.

+num l

Provoca o rolamento de tela do número "num" de linhas especificado. O rolamento pode se dar para frente(+) ou para trás (-). Se for usado um endereço absoluto, a impressão começará na linha especificada.

+num{**d**, <CTRL>D}

Avança ou retrocede o número "num" de meias páginas.

{**.**, <CTRL>L}

Reimprime a página corrente.

$

Mostra a última página do arquivo corrente.

PROCURA DE EXPRESSÕES

Os comandos, abaixo, são utilizados para procurar expressões-regulares em textos. As expressões-regulares descritas no comando **egrep**, são aceitas e devem terminar com <new line>, mesmo usando a opção **-n**.

num/exp/

Procura pela num-ésima ocorrência de "exp". A busca começa imediatamente depois da página corrente até o fim do arquivo.

num^exp^ ou num?exp?

Procura pela num-ésima ocorrência de "exp". A busca começa imediatamente antes da página corrente até o início.

Após a busca, **pg** apresenta a linha encontrada no topo da tela. Isto pode ser modificado se colocado no fim do comando **m** ou **b**, para que a saída da linha encontrada se faça no meio (**m**) ou no rodapé (**b**) da janela. O sufixo **t** pode ser usado para restaurar a situação original.

MODIFICA O AMBIENTE

Os comandos, a seguir, permitem ao usuário modificar o ambiente.

numn

Começa a análise a partir do num-ésimo arquivo posterior da linha de comando. O valor de **num** não deve conter sinal. Por omissão, num=1.

nump

Começa a análise a partir do num-ésimo arquivo anterior da linha de comando. O valor de **num** não deve conter sinal. Por omissão, num=1.

num

Mostra uma outra janela (do texto) de tamanho **num**, caso este seja especificado.

s arq_saída

Salva a entrada no arquivo arq_saída. Somente o arquivo corrente é salvo. Pode haver, ou não, um espaço entre **s** e **arq** e o comando deve terminar com um <new line>, mesmo que a opção **-n** esteja presente.

h

Oferece um resumo dos comandos disponíveis.

q ou **Q**

Finaliza o comando **pg**.

!comando

O argumento "comando" é enviado ao interpretador de comandos, cujo nome é obtido da variável de ambiente SHELL. Caso esta não esteja disponível, o interpretador default é assumido. Este comando deve sempre terminar por <new line>,mesmo que a opção **-n** esteja presente.

Sempre que a saída de **pg** estiver sendo enviada para o terminal, o usuário poderá utilizar as teclas **QUIT** ou **BREAK**, para que o comando deixe de de enviar textos para a saída e mostre, imediatamente, o "pronto" permitindo o uso dos comandos descritos anteriormente. Este procedimento pode causar a perda de algum texto que estava em fila para ser mostrado no vídeo, no momento do QUIT.

Se a saída não for o terminal, **pg** atua como se fosse o comando **cat**, com exceção de que o cabeçalho antecederá cada arquivo.

DIÁLOGO

Não existe.

CÓDIGOS DE RETORNO

zero - término sem erro.

diferente de zero - término com erro.

OBSERVAÇÕES

- Se os tabs do terminal não estiverem setados para oito posições, podem ocorrer resultados indesejáveis.
- Quando **pg** é usado como um filtro de outro comando que altere as opções de entrada e saída do terminal, a configuração do terminal pode não ser restaurada corretamente.

ARQUIVOS UTILIZADOS

/usr/lib/terminfo/*/*.

VEJA TAMBÉM

ed (comando)
grep (comando)

EXEMPLOS

. pg -ces -p/usr/lose/fontes/con*.c

Lista os arquivos-fonte "C" com prefixo "con", em páginas de dez linhas, com mensagens e pronto (em vídeo reverso), sem pausa no fim de cada arquivo, colocando o cursor na sua primeira posição e limpando a tela antes da impressão.

. pg -p%d -10 +/main/ /usr/lose/fontes/con*.c

Lista arquivos fonte "C" com prefixo "con", em páginas de dez linhas, numerando-as e iniciando no primeiro main do primeiro arquivo.

. pg -px +10 /usr/lose/fontes/con*.c

Lista os arquivos-fonte "C" de prefixo "con" em páginas de 23 linhas, iniciando pela décima linha do primeiro arquivo.

Página intencionalmente em branco.

PR **Comando - Básico**

pr - imprime arquivos.

SINTAXE

```
pr [-a] [-m] [-dfprt] [-coli] [-ecar1pos1]
   [-hnome] [-icar2pos2] [-lpag] [-ncar3col2]
   [-omarg] [+pag_inicial] [-scar4] [-wtam]  arq...
```

DESCRIÇÃO

O comando **pr** imprime, na saída-padrão, o conteúdo dos arquivos especificados. A listagem é dividida em páginas, contendo cada uma, um cabeçalho com o número da página, data, hora e o nome do arquivo.

Por default, o conteúdo dos arquivos é apresentado em colunas de mesmo tamanho, separadas por, pelo menos, um espaço em branco. As linhas maiores que as colunas são truncadas. Se a opção **-s** for usada, as linhas não serão truncadas e as colunas serão separadas por um caractere de separação.

OPÇÕES

O comando **pr** sem opções, gera uma saída contendo cabeçalho, o conteúdo do arquivo e um rodapé de linhas em brancos.

As opções a seguir, podem ser usadas individualmente ou combinadas em qualquer ordem:

-a

Produz a saída em várias colunas na página. Esta opção deve ser usada junto com a opção (-coll).

-m

Imprime simultaneamente, os arquivos especificados, um por coluna. Esta opção sobrepõe-se a opção (-coll).

-d

Produz a saída com espaçamento duplo de linhas, inserindo uma linha em branco entre as linhas com conteúdo.

-f

Usa o caractere **form-feed** para separar páginas (o default é uma seqüência de caracteres line feeds). Dá uma pausa antes de listar a primeira página, esperando um **carriage return**, caso a saída esteja associada ao terminal.

-p

Dá uma pausa antes de começar cada página, se a saída for um terminal, ou um beep e espera o acionamento da tecla **carriage return.**

-r

Não envia mensagens em caso de erro.

-t

Suprime as cinco primeiras linhas do cabeçalho e as cinco linhas em branco do rodapé para cada página.

-coll

Produz uma saída contendo coll colunas. Se omitido é assumido 1. Esta opção não deve ser usada com a opção **-m**. As opções **-e** e **-i** são assumidas para saídas em multicolunas.

-e

Substitui o carl por espaços em brancos, da seguinte forma: a cada ocorrência do caractere carl na linha, obtém-se sua posição e calcula-se cada elemento da sucessão: pos1+1, 2*pos1+1, 3*pos1+1, ..., até que se encontre um valor maior que a posição do caractere. Depois começa-se a inserir brancos, a partir da posição do caractere, até (exclusive) a posição representada pelo valor do elemento da sucessão.

-h

Substitui, no cabeçalho, o nome do arquivo pelo nome especificado.

-i

Substitui os caracteres em branco na saída, inserindo tabs para as posições pos2+1, 2*pos2+1, 3*pos2+1, ..., sempre que possível.

-l

Estipula o tamanho da página (número de linhas).

-n

Fornece uma numeração para cada linha com um número de dígitos ocupando as primeiras col2+1 posições para cada coluna da saída normal, ou cada linha da opção **-m**.

-o

Estabelece a indentação (em caracteres) para cada linha. O número de caracteres por linha corresponde à soma da indentação com o conteúdo das linhas.

+ pag_inicial

Especifica o número da página a partir da qual a impressão deve começar.

-s

Separa as colunas com o caractere definido por car4, ao invés do número apropriado de brancos (o valor de car4 por omissão é **TAB**).

-w

Estipula o tamanho da linha para "tam" caracteres, nas saídas em multicolunas. Por omissão, tam=72.

ARGUMENTOS

col1

Inteiro que especifica o número de colunas por página.

car1

Caractere a ser substituído por brancos na opção **-e**. Se omitido é assumido o caractere **TAB**.

pos1

Inteiro usado para obter a posição do elemento de sucessão. Se for omitido ou igual a zero será assumido 8.

nome

Cadeia de caracteres que contém o nome que substituirá o nome do arquivo no cabeçalho.

car2

Caractere, exceto dígito, a ser substituído pelo(s) caractere(s) branco. Se omitido será assumido o caractere **TAB**.

pos2

Inteiro usado para obter a posição do elemento de sucessão. Se for omitido ou igual a zero será assumido 8.

pag

Inteiro que especifica o número de linhas que contêm a página. Se omitido é assumido 66.

car3

Caractere, exceto dígito, usado para separar a numeração da linha e o seu conteúdo. Se omitido será assumido o caractere **TAB**.

col2

Inteiro que especifica o número de colunas reservadas para a numeração das linhas do arquivo. Se omitido é assumido 5.

marg

Especifica o número de caracteres usado na identação da linha. Se omitido é assumido zero.

pag_inicial

Inteiro maior que zero que epecifica a página a partir da qual começará a impressão. Se omitido será assumido zero.

car4

Caractere usado para separar as colunas (na opção **-s**). Se omitido é assumido o caractere **TAB**.

tam

Inteiro que especifica o número de caracteres por linha. Se omitido é assumido 72.

arq

Nome dos arquivos que se deseja imprimir. Se for omitido o nome de um arquivo ou for usado o caractere - (hífen) em seu lugar, a entrada-padrão será utilizada.

DIÁLOGO

Não existe.

CÓDIGOS DE RETORNO

zero - término sem erro.

diferente de zero - término com erro.

OBSERVAÇÕES

. Se o usuário desejar utilizar o comando **pr** no terminal, deve usar sempre as opções **-plk** de forma a controlar o rolamento de tela, já que o default para o tamanho da página é 66, referente à impressora.

. Caso a saída-padrão esteja associada ao terminal, as mensagens serão impressas após o término da listagem.

ARQUIVOS UTILIZADOS

Não existem.

EXEMPLO

```
pr -3drhRELAT arq1 arq2
```

Imprime arq1 e arq2, com espaçamento duplo, em 3 colunas e ao invés dos nomes dos arquivos, no cabeçalho, imprime RELAT. As mensagens de erro não serão enviadas.

PRS **Comando - Administração**

prs - imprime na saída-padrão um arquivo SCCS.

SINTAXE

```
prs [-a] [-e] [-l] [-c[data_hora]]
    [-d[pal_chave]] [-r$id] arq...
```

DESCRIÇÃO

O comando **prs** imprime na saída-padrão parte ou integralmente um arquivo SCCS com formato preestabelecido pelo usuário através dos argumentos.

Se um diretório é fornecido, **prs** pesquisa todos os arquivos deste diretório, exceto aqueles que não forem do tipo SCCS ou não tenham permissão de leitura, os quais serão ignorados.

Se - (hífen) é fornecido, é assumida a entrada-padrão. Cada linha desta, especifica o nome de um arquivo do tipo SCCS, ou um diretório a ser processado; um arquivo que não seja do tipo SCCS ou sem permissão de leitura é ignorado como visto anteriormente.

O comando **prs**, quando executado sem opções, fornece o SID mais recente do arquivo especificado.

Os argumentos podem aparecer em qualquer ordem e consistem de opções e nomes de arquivos.

OPÇÕES

-a

Lista informações sobre os deltas removidos (tipo = R), e existentes (tipo = D). Se a opção **-a** é omitida, só informações sobre os deltas existentes são listadas.

-e

Lista informações de todos os deltas criados antes (inclusive), do sid especificado ou data_hora nas opções **-r** ou **-c** respectivamente.

-l

Lista informações de todos os deltas criados depois (inclusive), do sid especificado ou data_hora nas opções **-r** ou **-c** respectivamente.

-c

Especifica a data e a hora limites para pesquisa dos deltas existentes.

-d

Permite listar parte do arquivo, segundo a palavra-chave especificada (pal_chave).

-r

Especifica a versão do arquivo a ser listada.

ARGUMENTOS

data_hora

Cadeia de caracteres que possui a seguinte sintaxe:

AA[MM[DD[HH[MM[SS]]]]]

onde cada par de caracteres significa,

AA - ano;
MM - mês;
DD - dia;
HH - horas;
MM - minutos;
SS - segundos.

Cada par descrito acima poderá ser (opcionalmente) separado por um caractere não-dígito. Para o par omitido é assumido o valor máximo.

pal_chave

Cadeia de caracteres que especifica as partes de um arquivo SCCS que devem ser listadas. Estas cadeias podem ser intercaladas por textos fornecidos pelo usuário. Não há limite para utilização de uma mesma palavra-chave numa opção **-d**. Ver item PALAVRA-CHAVE.

sid

Cadeia com até 19 caracteres que identifica a versão criada pelo comando **delta** , com o seguinte formato (ver comando **get**):

vvvv.nnnn ou vvvv.nnnn.rrrr.ssss

onde, v, n, r e s são inteiros.

PALAVRA-CHAVE

Todas as partes de um arquivo SCCS têm uma palavra-chave associada.

O formato de uma palavra-chave pode ser:

. simples(S) - onde a substituição da palavra-chave é direta.

. multilinhas(M) - onde a substituição da palavra-chave é seguida de um CR .

Uma tabulação, ou ordenação na tela, pode ser determinada por um **\t**, um **/** (carriage return) ou por **\n** (nova linha).

As palavras-chaves default são:

":DT:\t:DL:\nRMs:\n:MR:COMMENTS:\n:C:"

PALAVRAS CHAVES DE UM ARQUIVO SCCS

CHAVE	ITEM	SEÇÃO	VALOR	FORMATO
:Dt:	Informação do delta	(1)	(2)	S
:DL:	Estatísticas de linhas do delta	(1)	:Li:/:Ld:/:Lu:	S
:Li:	Linhas inseridas por delta	(1)	nnnnn	S
:Ld:	Linhas deletadas por delta	(1)	nnnnn	S
:Lu:	Linhas não alteradas por delta	(1)	nnnnn	S
:DT:	Tipo de delta	(1)	D ou R	S
:I:	SID	(1)	:R:.:L:.:B:.:S:	S
:R:	Número da versão	(1)	nnnnn	S
:L:	Número de nível	(1)	nnnnn	S
:B:	Número de desvio	(1)	nnnnn	S
:S:	Número de seqüência	(1)	nnnnn	S
:D:	Data de criação do delta	(1)	:Dy:/:Dm:/:Dd:	S
:Dy:	Ano de criação do delta	(1)	nn	S
:Dm:	Mês de criação do delta	(1)	nn	S
:Dd:	Dia de criação do delta	(1)	nn	S
:T:	Tempo de criação do delta	(1)	:Th:::Tm:::Ts:	S
:Th:	Hora de criação do delta	(1)	nn	S
:Tm:	Minutos de criação do delta	(1)	nn	S
:Ts:	Segundos de criação do delta	(1)	nn	S
:P:	Programador que criou o delta	(1)	nome da sessão	S
:DS:	Número de seqüência do delta	(1)	nnnn	S
:DP:	Número de seqüência do delta antecessor	(1)	nnnn	S
:DI:	Número de seqüência dos deltas incluídos, excluídos e ignorados	(1)	:Dn:/:Dx:/:Dg:	S
:Dn:	Deltas incluídos	(1)	:DS: :DS: ...	S
:Dx:	Deltas excluídos	(1)	:DS: :DS: ...	S
:Dg:	Deltas ignorados	(1)	:DS: :DS: ...	S
:MR:	Número RM para delta	(1)	texto	M
:C:	Comentários para delta	(1)	texto	M
:UN:	Nome da sessão	Nomes das Sessões	texto	M
:FL:	Lista de indicadores	(3)	texto	M
:Y:	Indicador de tipo do módulo	(3)	texto	S
:MF:	Indicador de validação de RM	(3)	SIM ou NÃO	S
:MP:	Nome do programa de validação de RM	(3)	texto	S
:KF:	Indicador erro/advertência de palavra chave	(3)	SIM ou NÃO	S
:KV:	Cadeia de validação de chave	(3)	texto	S
:BF:	Indicador de sessão	(3)	SIM ou NÃO	S
:J:	Indicador de edição de múltipla	(3)	SIM ou NÃO	S
:LK:	Liberação bloqueada	(3)	:R: ...	S
:Q:	Palavra chave de definição de usuário	(3)	texto	S
:M:	Nome do módulo	(3)	texto	S
:FB:	Limite mínimo	(3)	:R:	S
:CB:	Limite máximo	(3)	:R:	S
:Ds:	SID default	(3)	:I:	S
:ND:	Indicador delta nulo	(3)	SIM ou NÃO	S
:FD:	Texto de Arquivo Descritivo	Comentário	texto	M
:BD:	Corpo	Corpo	texto	M
:GB:	Corpo formado	Corpo	texto	M
:W:	Forma de cadeia conhecida pelo what	N/A	:Z::M:\t:I:	S
:A:	Cadeia what	N/A	:Z::Y: :M: :I::Z:	S
:Z:	Delimitador da cadeia what	N/A	@(#)	S
:F:	Nome do arquivo SCCS	N/A	texto	S
:PN:	Caminho do arquivo SCCS	N/A	texto	S

(1) Tabela de delta
(2) :Dt: = :DT: :I: :D: :T: :P: :DS: :DP:
(3) Indicadores

DIÁLOGO

Não existe.

CÓDIGOS DE RETORNO

zero - término sem erro.

diferente de zero - término com erro.

OBSERVAÇÃO

A informação impressa por **prs** consiste de:

a) Texto de suporte fornecido pelo usuário;

b) Valores apropriados (extraídos do arquivo SCCS) substituídos pelas palavras chaves na ordem de disposição no argumento pal_chave.

VEJA TAMBÉM

admin (comando)
delta (comando)
get (comando)
what (comando)

EXEMPLOS

```
.!prs -d "Dia - :Dd: \n Hora - :T:" s.arq

   Dia - 15
   Hora - 11:16:33
```

```
. prs -d :Dt: s.arq

  D 11 88/09/15 11:16:33 lys 1 0
```

Página intencionalmente em branco.

PS **Comando - Básico**

ps - informa estados dos processos ativos.

SINTAXE

```
ps [opções]
```

DESCRIÇÃO

O comando **ps** imprime informações sobre processos ativos. Se nenhuma opção for dada, informa sobre os processos associados ao terminal corrente. Neste caso, a saída consiste nas seguintes informações:

- identificador de processo (idp);
- identificador de terminal;
- tempo de execução acumulado;
- nome do comando sendo executado.

Fazendo uso das **opções**, maiores informações podem ser fornecidas.

As opções que usam lista de argumentos podem ter estas listas fornecidas em uma das duas formas abaixo:

- Os identificadores da lista separados entre si por vírgulas.

- Lista entre aspas duplas, com os identificadores separados por vírgula e/ou por um ou mais espaços.

OPÇÕES

-e

Imprime informações sobre todos os processos ativos.

-d

Imprime informações sobre todos os processos, com exceção dos processos líderes de grupo.

-a

Imprime informações sobre todos os processos, com exceção dos processos líderes de grupo e dos processos que não estão associados a nenhum terminal.

-t lista

Imprime informações apenas sobre os processos associados aos terminais especificados na **lista.** Existem duas maneiras de especificar os terminais:

- Nome do arquivo especial (por exemplo, tty4);
- Apenas o número (por exemplo, 4), se o nome do arquivo começar com **tty.**

-p lista

Imprime informações só sobre os processos cujos identificadores numéricos sejam dados na lista.

-u lista

Imprime informações apenas sobre os processos cujos identificadores de usuário, ou nomes de sessão estiverem especificados na **lista**. Na impressão, o identificador de usuário será mostrado, a não ser que a opção **-f** seja especificada. Nesse caso o nome de sessão é que será impresso.

-g lista

Imprime informações apenas sobre os processos cujos líderes de grupo estejam especificados na **lista**, através dos seus identificadores de processo.

-l

Produz uma listagem completa (ver "Observações").

-f

Produz uma listagem resumo da completa (ver "Observações").

ARGUMENTOS

Não existem.

DIÁLOGO

Não existe.

CÓDIGOS DE RETORNO

zero - término sem erro.

diferente de zero - término com erro.

OBSERVAÇÃO

Os títulos das colunas e seus significados são fornecidos a seguir. As letras entre parênteses indicam o tipo das informações listadas para as opções -f e -l. A palavra **todas** entre parênteses indica que aquela informação aparece em todos os tipos de listagem.

As informações são:

-F(l)

"Flag" associado ao processo: 01 carregado em memória
02 processo do sistema

S(l)

Estado do processo: S dormindo
W esperando
R executando
I intermediário
Z terminado
T parado

UID(f,l)

Identificador do usuário dono do processo, ou o nome da sessão se for a opção -f.

PID(todas)

Identificador de processo (é possível matar um processo conhecendo essa informação).

PPID(f,l)

Identificador do processo-pai.

C(f,l)

Utilização do processador para despacho.

PRI(l)

Prioridade do processo (quanto maior o número, menor será a prioridade do processo).

NI(l)

Valor "nice", usado no cálculo de prioridades.

ADDR(l)

Endereço físico de memória do segmento de código do processo, em unidades de koctetos.

SZ(l)

Tamanho, em unidades de 512 octetos, de toda a memória ocupada pelo processo.

WCHAN(l)

Evento pelo qual o processo está esperando ou dormindo; se for branco, o processo está executando.

STIME(f)

Hora do início da execução do processo.

TTY(todas)

Terminal de controle associado ao processo.

TIME(todas)

Tempo de execução acumulado do processo.

CMD(todas)

Nome do comando. O nome completo e seus argumentos são impressos se a opção **-f** for especificada.

- Se um processo terminou e tem um pai, mas este pai ainda não esperou pelo seu término, ele é marcado como **zumbi**.

- A situação pode mudar enquanto **ps** está executando. A imagem que ele fornece é apenas uma aproximação da realidade, sem possuir a garantia de ser um "instântaneo" do sistema.

ARQUIVOS UTILIZADOS

/dev/kmem
/etc/passwd
/etc/utmp

PWCK **Comando - Administração**

pwck - consiste um arquivo de senhas.

SINTAXE

```
/etc/pwck [arq]
```

DESCRIÇÃO

O comando **pwck** verifica todas as entradas no arquivo de senhas.

Esta verificação consiste na validação do número de campos por entrada, do nome de abertura de sessão, do identificador de usuário, do identificador de grupo e, se estiverem especificados, do diretório de abertura de sessão e do programa opcional.

OPÇÕES

Não existem.

ARGUMENTOS

arq

Arquivo de senhas a ser verificado. Se omitido, assume o arquivo /etc/passwd.

DIALOGO

Não existe.

CÓDIGOS DE RETORNO

zero - término sem erro.

diferente de zero - término com erro.

ARQUIVOS UTILIZADOS

/etc/group
/etc/passwd

PWD **Comando - Básico**

pwd - imprime o diretório corrente.

SINTAXE

```
pwd
```

DESCRIÇÃO

O comando **pwd** imprime o caminho do diretório corrente.

OPÇÕES

Não existem.

ARGUMENTOS

Não existem.

DIÁLOGO

Não existe.

CÓDIGOS DE RETORNO

zero - término sem erro.

diferente de zero - término com erro.

VEJA TAMBÉM

cd (comando)

RECUTRA **Comando - Administração**

recutra - recupera transação.

SINTAXE

```
recutra [nome_do_arquivo_de_histórico]
```

DESCRIÇÃO

O comando **recutra** tem por finalidade prover o Método de Acesso Seqüencial Indexado do Sistema de Arquivamento, bem como a Biblioteca C-ISAM, de uma ferramenta de recuperação, após erros causados por queda de força ou disco defeituoso, durante processamento de transações.

OPÇÕES

Não existem.

ARGUMENTOS

nome_do_arquivo_de_histórico

Nome do arquivo que deve fornecer os dados de recuperação da transação.

DIÁLOGO

Não existe.

CÓDIGOS DE RETORNO

zero - término sem erro.

diferente de zero - término com erro.

OBSERVAÇÕES

- A recuperação é feita a partir do arquivo de histórico e de uma cópia de segurança dos arquivos envolvidos na transação, sendo esperado que o usuário restaure o arquivo a partir da cópia de segurança, que os diretórios e volumes estejam montados de modo a reproduzir o ambiente corrente ao início das transações, e que o arquivo de histórico fornecido reflita todas as atualizações feitas após essa versão do arquivo de segurança.

- É aconselhável que, durante a transação, sejam utilizadas as expressões de caminho completas para os arquivos envolvidos.

- O comando **recutra** não faz parte do pacote do SOX. É fornecido, separadamente, pela COBRA, junto com o sistema de arquivamento estendido (SAX).

ARQUIVOS UTILIZADOS

O utilitário consulta o arquivo de histórico de modo a obter os dados de recuperação e atualiza os arquivos envolvidos na transação bem sucedida, recuperados a partir das respectivas cópias de segurança.

EXEMPLOS

a) Operação bem sucedida:

```
. recutra historico
```

b) Operação mal sucedida

```
. recutra historico

  recutra : Fatal em abertura de arquivo
            Arq1 não foi aberto para atualização
```

Página intencionalmente em branco.

RED **Comando - Básico**

red - editor de texto.

SINTAXE

```
red [-][-s]   [-p novopronto] [-d] [arq]
```

DESCRIÇÃO

O utilitário **red** é um editor de textos ou seja, um programa interativo que permite criar ou alterar textos armazenados em arquivos.

OPÇÕES

Ver o comando **ed.**

ARGUMENTOS

Ver o comando **ed.**

DIÁLOGO

Não existem.

CÓDIGOS DE RETORNO

Não existem.

OBSERVAÇÃO

O comando **red** é uma versão resumida do comando **ed.** Ele somente permite a edição de arquivos presentes no diretório corrente e não eceita a execução de outros utilitários do seu interior (**!** comando **sox**).

Os demais comandos são idênticos aos do comando **ed.**

ARQUIVOS UTILIZADOS

Não existem.

VEJA TAMBÉM

ed (comando)
grep (comando)

EXEMPLO

Ver o comando **ed.**

RM **Comando - Básico**

rm - remove arquivos ou diretórios.

SINTAXE

```
rm [-fri] arq...
```

DESCRIÇÃO

O comando **rm** remove as entradas de um ou mais arquivos de um diretório. Se um arquivo possui apenas uma associação no diretório, **rm** destrói o arquivo.

Para uma entrada de um arquivo ser removida, o diretório ao qual pertence esta entrada deve ter permissão de escrita (essa restrição não se aplica a superusuários), porém, não requer permissão de leitura ou escrita no arquivo em questão.

Se um arquivo não puder ser removido, **rm** enviará uma mensagem de erro e continuará executando normalmente. Ao terminar devolve como código de retorno o total de arquivos não removidos (o valor retornado é módulo 101).

Se um arquivo não possui permissão de escrita e a entrada-padrão é um terminal, o comando imprime a permissão do arquivo e uma linha é lida dessa entrada.

Se a linha começa com um dos caracteres do conjunto "sSyY", o arquivo é removido, caso contrário ele é mantido. Nenhuma pergunta é feita se a opção **-f** é usada, ou se a entrada-padrão não é um terminal.

Uma mensagem de erro é emitida se o comando **rm** for usado, sem a opção **-r**, para remover um diretório.

A opção **-r** permite remover recursivamente as entradas de um arquivo-diretório e o próprio diretório dado.

Se a opção **-i** é usada, **rm** pergunta se deve remover cada arquivo, e com **-r** se deve percorrer cada diretório, antes de fazê-lo.

OPÇÕES

-f

Omite a pergunta, no caso do arquivo não possuir permissão de escrita e a entrada-padrão ser um terminal. Assume que a resposta é sim.

-i

Pergunta, para cada arquivo, se deseja removê-lo, e com **-f** também se deseja examinar cada diretório.

-r

Remove recursivamente as entradas de um diretório e por fim ele próprio.

ARGUMENTO

arq

Nome do arquivo ou do diretório a ser removido.

DIÁLOGO

Não existe.

CÓDIGOS DE RETORNO

zero	- término sem erro.
de 1 a 100	- total de arquivos não removidos.
maior que 100	- término com erro.

OBSERVAÇÕES

. Os arquivos "." e ".." não podem ser removidos, para evitar as conseqüências de algo como rm -r .* .

. O comando rm não remove diretórios, a menos que a opção "-r" seja usada.

ARQUIVOS UTILIZADOS

Não existem.

VEJA TAMBÉM

unlink (chamada)
rmdir (comando)

RMDEL **Comando - Desenvolvimento**

rmdel - remove o delta de arquivo SCCS.

SINTAXE

```
rmdel -rsid arq...
```

DESCRIÇÃO

O comando **rmdel** remove o delta especificado de cada arquivo SCCS. O delta a ser removido deve ser o mais recente na ramificação. O sid especificado não pode ser o mesmo na versão em edição para posterior execução do comando delta, isto é, se existir um arquivo tipo "p" (vide comando **get**), relacionado ao arquivo SCCS passado no comando **rmdel**, o sid especificado não deverá constar do conteúdo desse arquivo tipo "p".

Se um diretório é passado como argumento, **rmdel** se comporta como se cada arquivo no diretório tivesse sido especificado, exceto os que não são do tipo SCCS (aqueles em que o último componente do "nome do caminho" não comece por **s.**). Os arquivos sem permissão de leitura são ignorados.

Se - (hífen) for passado como nome de arquivo, assume-se a entrada-padrão, e cada linha é assumida como nome de um arquivo SCCS.

A remoção de um delta poderá ser feita pelo usuário que o criou, o dono do arquivo ou o dono do diretório.

OPÇÃO

-r

Remove o delta especificado do(s) arquivo(s) SCCS.

ARGUMENTOS

sid

Cadeia com até 19 caracteres que identifica a versão criada pelo comando **delta**, com o seguinte formato (ver comando **get**):

vvvv.nnnn ou vvvv.nnnn.rrrr.ssss

onde, v, n, r e s são inteiros.

arq

Arquivo tipo SCCS ou diretório a serem removidos.

DIÁLOGO

Não existe.

CÓDIGOS DE RETORNO

zero - término sem erro.

diferente de zero - término com erro.

OBSERVAÇÃO

Nas mensagens de advertência e apoio, a versão (modificação) de um arquivo-fonte é antecedida pela expressão "Delta". Exemplo: rmdel : Delta 1.2 ... .

ARQUIVOS UTILIZADOS

Não existem.

VEJA TAMBÉM

delta (comando)
get (comando)
prs (comando)

EXEMPLO

. rmdel -r1.3 /usr/carlos

São removidos os deltas 1.3 de todos os arquivos que comecem por **s.** do diretório /usr/carlos.

Página intencionalmente em branco.

RMDIR **Comando - Básico**

rmdir - remove diretórios.

SINTAXE

```
rmdir dir ...
```

DESCRIÇÃO

O comando **rmdir** remove os diretórios especificados. O diretório deve estar vazio, exceto pelas entradas "." e "..", que ele próprio remove.

OPÇÕES

Não existem.

ARGUMENTO

dir

Diretório a ser removido.

DIÁLOGO

Não existe.

CÓDIGOS DE RETORNO

zero - término sem erro.

diferente de zero - término com erro.

OBSERVAÇÃO

Não é permitida a remoção do diretório-corrente, do diretório ".." e de diretórios com volumes montados.

ARQUIVOS UTILIZADOS

Não existem.

VEJA TAMBÉM

rm (comando)

SACT **Comando - Desenvolvimento**

sact - lista as versões dos arquivos tipo SCCS editados.

SINTAXE

```
sact arq...
```

DESCRIÇÃO

O comando **sact** fornece ao usuário informações (SID, data de criação, etc..) das versões do(s) arquivo(s) SCCS em que o comando **get** foi executado com a opção **-e**, sem a posterior execução do comando **delta.**

Se um diretório foi passado como argumento, todos os arquivos desse diretório são processados excetuando-se os arquivos que não sejam SCCS e os arquivos sem permissão para leitura. Se a opção - (hífen) foi indicada, cada linha da entrada-padrão é lida como o nome de um arquivo ou diretório a ser processado.

A sintaxe fornecida sobre cada arquivo apresenta o seguinte formato:

nome do arquivo: campo1 campo2 campo3 campo4 campo5

onde,

campo 1: especifica o SID do delta existente no arquivo SCCS "**s.**nome do arquivo", na qual serão feitas as alterações para a criação do novo **delta**;

campo 2: especifica o SID do delta a ser criado, contido no arquivo "**p.**nome do arquivo". Esta será retirada da lista ao ser executado o comado **delta** e passará a fazer parte do arquivo "**s.**nome do arquivo";

campo 3: contém o "nome da sessão" do usuário que fará um nova modificação no arquivo fonte, isto é, executou um **get** para edição;

campo 4: contém a data de execução do comando **get** com opção **-e**;

campo 5: contém a hora de execução do comando **get** com opção **-e.**

OPÇÕES

Não existem.

ARGUMENTO

arq

Especifica o nome do arquivo tipo SCCS ou diretório a ser tratado pelo comando.

DIÁLOGO

Não existe.

CÓDIGOS DE RETORNO

zero - término sem erro.

diferente de zero - término com erro.

ARQUIVOS UTILIZADOS

Não existem.

VEJA TAMBÉM

get (comando)
delta (comando)
unget (comando)

EXEMPLO

. sact s.b

```
s.b:    1.1 1.2 mineiro 88/09/13 10:50:47
        1.1 1.1.1.1 mineiro 88/09/17 15:55:20
```

Através do comando **get** com a opção **-e** foram criadas duas versões (1.2 e 1.1.1.1) para o arquivo s.b. Essas versões passarão a fazer parte do arquivo s.b após a execução do comando **delta**. Não constarão da lista quando executado o próximo comando **sact**.

Página intencionalmente em branco.

SETMNT **Comando - Administração**

setmnt - cria tabela de volumes montados.

SINTAXE

```
/etc/setmnt
```

DESCRIÇÃO

O comando **setmnt** cria o arquivo /etc/mnttab (tabela de volumes montados), que é usada pelos comandos **mount** e **umount**. O comando **setmnt** lê a entrada-padrão e cria uma entrada na tabela para cada linha. As linhas de entrada têm o seguinte formato:

arqesp arqdir

onde arqesp é o nome do arquivo especial no diretório /dev associado ao volume e arqdir é o diretório associado à raiz daquele volume. Então arqesp e arqdir tornam-se as primeiras duas cadeias na entrada de /etc/mnttab.

OPÇÕES

Não existem.

ARGUMENTOS

Não existem.

DIÁLOGO

Não existe.

CÓDIGOS DE RETORNO

zero - término sem erro.

diferente de zero - término com erro.

ARQUIVOS UTILIZADOS

/etc/mnttab

VEJA TAMBÉM

mount (comando)

SFO **Comando - Administração**

sfo - finaliza o sistema.

SINTAXE

```
/etc/sfo
```

DESCRIÇÃO

O comando **sfo** faz parte das rotinas de operação do sistema. Sua função é desativar o SOX de uma forma ordenada.

O comando **sfo** causa o cancelamento de todos os processos em curso no sistema e avisa ao processo **init** que o sistema está saindo de operação.

As conseqüências da desativação do sistema são as seguintes:

- Todas as áreas de disco que estejam em memória, são atualizadas (equivalente ao **sync**).
- Todos os volumes montados (inclusive o volume raiz) são desmontados (equivalente ao comando **umount**).
- O sistema sai de operação, só retornando através de uma nova ativação.

Este comando só pode ser executado pelo superusuário.

OPÇÕES

Não existem.

ARGUMENTOS

Não existem.

DIÁLOGO

Não existe.

CÓDIGOS DE RETORNO

Não existem.

OBSERVAÇÕES

- É obrigatória a execução desse comando como última ação, antes da máquina ser desligada ou teclar **RESET.**
- Recomenda-se que o comando **sfo** não seja utilizado isoladamente, mas através do arquivo de comando **shutdown.**

ARQUIVOS UTILIZADOS

Não existem.

VEJA TAMBÉM

fsck (comando)
shutdown (comando)
sync (comando)
umount (comando)

Página intencionalmente em branco.

SH,RSH **Comando - Básico**

sh,rsh - interpretador de comandos padrão/restrito.

SINTAXE

```
sh   [opções]  [arg]
rsh  [opções]  [arg]
```

DESCRIÇÃO

O comando **sh** é um interpretador de comandos que executa comandos lidos de um terminal ou de um arquivo.

O comando **rsh** é uma versão restrita do comando **sh**. Uma Shell restrita não permite:

- Mudanças de diretório-corrente.
- Modificar a variável ambiental PATH.
- Especificar caminhos ou comandos contendo "/".
- Fazer redireções da saída-padrão (> e >>).

A ativação de **sh** sem opções ou argumentos, indica que os comandos serão lidos da entrada-padrão.

A seguir são apresentadas as opções e argumentos da sintaxe da **sh** e **rsh.** O Manual de Referência da Linguagem Shell/Sox apresenta uma descrição completa de todas as facilidades oferecidas pela Shell.

OPÇÕES

As opções abaixo são interpretadas pela Shell somente no momento de sua ativação. Note que, a menos que as opções **-c** ou **-s** sejam especificadas, o primeiro argumento deve ser um arquivo de comandos e os argumentos restantes são os parâmetros posicionais deste arquivo.

-c cadeia

Se esta opção estiver presente, os comandos são lidos de **cadeia** e executados.

-s

Se esta opção estiver presente, ou se não restarem mais argumentos, os comandos são lidos da entrada-padrão. Quaisquer argumentos restantes especificam os parâmetros posicionais. As saídas da Shell (exceto para comandos internos) são escritas no arquivo de descritor número 2 (dois).

-i

Se esta opção estiver presente ou se a entrada e a saída da Shell estiverem associadas ao terminal, esta Shell é dita interativa. Neste caso o sinal SIGTERM é ignorado (o comando **kill** com argumento "0" não matará uma Shell interativa), e o sinal SIGINT e' capturado e ignorado (assim **wait** poderá ser interrompido). Em todos os casos, o sinal SIGQUIT será ignorado pela Shell.

-r

Determina uma Shell restrita.

ARGUMENTO

arg

Arquivo de comandos ou parâmetro posicional.

DIÁLOGO

Não existe.

CÓDIGOS DE RETORNO

zero - término sem erro.

diferente de zero - término com erro.

ARQUIVOS UTILIZADOS

/dev/null
/etc/profile
$HOME/.profile.

VEJA TAMBÉM

Manual de Referência Linguagem Shell/SOX (LE01)

SHUTDOWN **Comando - Administração**

shutdown - desativa sistema.

SINTAXE

/etc/shutdown

DESCRIÇÃO

O comando **shutdown** corresponde a um arquivo de comandos para o interpretador padrão (Shell). Sua função é desativar o sistema gradual e ordenadamente.

O conteúdo deste arquivo não é padronizado. Ele deve conter os comandos que o administrador julgar necessários para a desativação do sistema.

As ações que devem ser tomadas através de comandos do arquivo **shutdown** são:

- Avisar a todos os usuários ativos no sistema, que o mesmo será desativado dentro de algum tempo (por exemplo, via comando **wall**).
- Aguardar algum tempo, para que os usuários suspendam a execução de seus processos;
- Desativar o sistema (por exemplo, via comando **sfo**).

OPÇÕES

Não existem.

ARGUMENTOS

Não existem.

DIÁLOGO

Não existe.

CÓDIGOS DE RETORNO

Não existem.

OBSERVAÇÃO

Deve ser lembrado que é obrigatória a execução do comando **sfo**, antes de se desligar a máquina ou teclar **RESET**. Portanto, **sfo** deve ser sempre incluído como o último comando deste arquivo.

ARQUIVO UTILIZADO

/etc/shutdown

VEJA TAMBÉM

sleep (comando)
sfo (comando)
wall (comando)
Manual do Administrador do Sistema SOX (SX03)

Página intencionalmente em branco.

SIZE **Comando - Desenvolvimento**

size - imprime tamanho das seções de arquivos-objeto.

SINTAXE

```
size   [-o][-x][-V]   arq...
```

DESCRIÇÃO

O comando **size** imprime, para cada arquivo dado, o tamanho de cada seção do arquivo-objeto em questão, seguido da soma dos tamanhos de suas seções.

Se uma biblioteca for passada como argumento, a mesma informação será impressa para cada um de seus membros.

OPÇÕES

Sem opções, o comando **size** lista os tamanhos das seções em decimal.

-o

Imprime os valores numéricos em octal.

-x

Imprime os valores numéricos em hexadecimal.

-V

Imprime a versão do comando **size** que está sendo executado na saída-padrão de erros.

ARGUMENTO

arq

Arquivo-objeto ou biblioteca construída pelo comando **ar**.

DIÁLOGO

Não existe.

CÓDIGOS DE RETORNO

zero - término sem erro.

diferente de zero - término com erro.

ARQUIVOS UTILIZADOS

Não existem.

VEJA TAMBÉM

ar (comando)
cc (comando)
ld (comando)

SLEEP **Comando - Básico**

sleep - suspende a execução de um processo por um tempo determinado.

SINTAXE

```
sleep  segundos
```

DESCRIÇÃO

O comando **sleep** causa um atraso na execução do processo, de tantos segundos quantos forem especificados no argumento segundos.

OPÇÕES

Não existem.

ARGUMENTOS

segundos

Número de segundos.

DIÁLOGO

Não existe.

CÓDIGO DE RETORNO

zero - término sem erro.

ARQUIVOS UTILIZADOS

Não existem.

VEJA TAMBÉM

alarm (chamada)
sleep (chamada)

EXEMPLOS

. (sleep 60; comando) &
Atrasa a execução do comando de 1 (um) minuto.

```
. while true
  do
    comando
    sleep 25
  done
```
Executa comando de 25 em 25 segundos.

SORT **Comando - Básico**

sort - classifica e/ou intercala arquivos.

SINTAXE

```
sort [-bcdfimnru] [-oarq_saida] [-ttar]
     [-ymem] [-ztam] [+pos1] [-pos2] [arq... ]
```

DESCRIÇÃO

O comando **sort** classifica/intercala as linhas dos arquivos de entrada e imprime o resultado na saída-padrão, ou no arquivo especificado na opção **-o**arq_saida.

A entrada-padrão é lida se o caractere - (hífen) for usado no lugar do nome de arquivo ou se nenhum arquivo de entrada for especificado.

As comparações são baseadas em uma ou mais chaves de classificação extraídas de cada linha de entrada. Por default a chave de classificação corresponde à linha de entrada (inteira) e a ordenação é lexicográfica pelos octetos da seqüência de intercalação da máquina.

OPÇÕES

-b

Ignora os primeiros brancos quando determinam a posição inicial e final de uma chave de classificação. Se a opção -b for especificada antes do primeiro argumento +pos1, ela será aplicada para todos os argumentos +pos1. Por outro lado, o indicador "b" pode ser vinculado, independentemente, para cada argumento +pos1 ou -pos2 (pos1 e pos2).

-c

Verifica se o arquivo de entrada está ordenado. Não gera nenhuma saída, caso o arquivo esteja desordenado.

-d

Considera somente letras, dígitos e brancos (espaços e TABS) nas comparações.

-f

Transforma as letras minúsculas em letras maiúsculas para a comparação dos campos. Mantém a forma original na saída. Esta opção deve ser usada para ordenar arquivos que contenham letras maiúsculas e minúsculas.

-i

Ignora os caracteres fora do padrão ASCII 040-0176 nas comparações não numéricas.

-m

Faz somente a intercalação dos arquivos de entrada. Os arquivos de entrada devem estar ordenados.

-n

Executa comparações numéricas. Uma cadeia de dígito, pode ter no início espaços em branco opcionais, sinal de menos opcional e zero ou mais dígitos com ponto decimal opcional. A ordenação é de acordo com o valor aritmético. A opção **-n** subentende a opção **-b** . Note que a opção **-b** somente se efetiva quando se especifica chaves de classificação.

-r

Organiza os itens em ordem decrescente. Normalmente os itens são classificados em ordem crescente.

-u

Suprime linhas repetidas com exceção de uma em cada conjunto de linhas com chaves iguais.

-o

Permite alterar a saída-padrão para o arquivo especificado (arq_saida).

-t

Especifica o caractere delimitador de campo.

-y

Permite ao usuário especificar o espaço de memória que o **sort** deve utilizar, pois este tem um grande impacto na sua performance. Ordenar um arquivo pequeno num grande espaço de memória, é um desperdício. Se esta opção for omitida, o comando **sort** inicia usando o espaço default, e se necessário, utiliza mais espaço de memória.

-z

Esta opção permite ao usuário especificar o tamanho da maior linha lida. O tamanho da maior linha lida é gravada na fase de classificação, assim buffers podem ser alocados durante a fase de intercalação. Se a fase de classificação é omitida, através das opções **-c** ou **-m**, será usado um tamanho default. Linhas maiores que o tamanho do buffer farão com que **sort** termine anormalmente.

ARGUMENTOS

arq_saida

Este argumento especifica o arquivo a ser usado como saída-padrão. Este arquivo pode ser um dos arquivos de entrada. É opcional o uso de brancos entre "-o" e "arq_saida".

car

Caractere usado como delimitador de campos no arquivo de entrada. Não é considerado parte do campo, embora possa ser incluído na chave de classificação. Cada ocorrência de car é significativa (por exemplo para **-t**x, "xx" delimita um campo vazio").

mem

Inteiro que especifica (em kocteto) o espaço de memória necessário para a execução do comando. **sort.** Se esta opção for especificada, o **sort** começará usando aquele número de "mem" octetos, a menos que o número mínimo ou máximo seja violado. Neste caso, o correspondente extremo será usado. Com o uso de "-y0" é garantida a memória mínima para iniciar. Por convenção, "-y" (sem argumento) inicia com máximo de memória.

tam

Inteiro que fornece o número real de octetos da maior linha a ser intercalada que impede o término anormal do **sort.**

pos1 e pos2

Especificam uma chave de classificação iniciando em pos1 e finalizando em pos2. Os caracteres nas posições pos1 e pos2 são incluídos na chave de classificação (desde que pos2 não preceda pos1). A ausência de -pos2 significa o fim da linha.

A especificação de pos1 e pos2 envolve a noção de um campo, uma seqüência mínima de caracteres seguidos por um separador de campos ou um nova linha. Por default, o primeiro branco (espaço ou "TAB") de uma seqüência de brancos age como o separador de campos.

Todos os brancos em uma seqüência de brancos são considerados sendo parte do próximo campo; por exemplo, todos os brancos do início de uma linha são considerados como sendo parte do primeiro campo.

Os argumentos +pos1 e -pos2 têm cada um a forma m.n opcionalmente seguida por uma ou mais opções **b, d, f, i, n** e **r.** A posição inicial especificada por +m.n é interpretada como (n+1) ésimo caractere do (m+1) ésimo campo. A falta de n significa .0, indicando o primeiro caractere do (m+1) ésimo campo. Se o indicador "b" for especificado, "n" é contado do primeiro não-branco do (m+1) ésimo campo; +m.0b referencia o primeiro caractere não-branco do (m+1) ésimo campo.

A última posição especificada por m.n é interpretada para significar o (n) ésimo caractere (incluindo separadores) depois do último caractere do (m) ésimo campo. A omissão de .n significa 0, indicando o último caractere do (m) ésimo campo. Se o indicador "b" for especificado, "n" é contado a partir do último branco do (m+1) ésimo campo; -m.1b refere-se ao primeiro caractere não-branco no (m+1) ésimo campo.

arq

Nomes dos arquivos que se deseja ordenar/intercalar.

DIÁLOGO

Não existe.

CÓDIGOS DE RETORNO

zero - término sem erro.

diferente de zero - término com erro.

OBSERVAÇÕES

. Quando as opções aparecem antes das chaves de classificação, as regras de ordenação pedidas são aplicadas globalmente para todas as chaves de classificação. Quando vinculada a uma chave específica de classificação (descrita abaixo), a opção de ordenação especifica sobrepõe-se a toda ordenação global por esta chave.

. Quando na última linha do arquivo faltar o caractere "**new line**", **sort** coloca em new line, envia mensagem de advertência e continua execução.

. O comando **sort** envia comentários e retorna código não-zero para várias situações anormais, e para arquivos fora de ordem, no caso do uso da opção **-c**.

. Quando existem múltiplas chaves de classificação, chaves posteriores são comparadas apenas quando todas as chaves anteriores forem iguais. Linhas iguais entretanto, são ordenadas através de todos os octetos significativos.

ARQUIVOS UTILIZADOS

Não existem.

VEJA TAMBÉM

comm (comando)
join (comando)
uniq (comando)

EXEMPLO

Para o arquivo cujo conteúdo é:

```
Maria f p 19
Sonia f p 6
Juliana f p 18
```

com a utilização do **sort** com as seguintes opções:

```
. sort -nr +3 -4 arq
```

obtemos:

```
Maria f p 19
Juliana f p 18
Sonia f p 6
```

SPLIT **Comando - Básico**

split - particiona arquivo.

SINTAXE

```
split [-n] [arq [nome]]
```

DESCRIÇÃO

O comando **split** lê do arquivo de origem e o grava em partes de **n** linhas (default 1000 linhas) em um conjunto de arquivos de saída.

Os nomes dos arquivos de saída são gerados a partir da concatenação do **nome** fornecido com os sufixos de "aa" a "zz", em ordem lexicográfica (o número máximo é, portanto, de 676 arquivos).

OPÇÕES

Não existem.

ARGUMENTOS

-n

n é o número de linhas a serem gravadas nos arquivos de saída. Na sua ausência assume 1000.

arq

Arquivo que deve ser lido (lê da entrada-padrão por omissão, ou se esse argumento for -).

nome

Nome que será utilizado pelo utilitário para gerar os arquivos de saida. O default é **x**.

DIÁLOGO

Não existe.

CÓDIGOS DE RETORNO

zero - término sem erro.

diferente de zero - término com erro.

OBSERVAÇÃO

O argumento nome não pode ter mais de 12 caracteres.

ARQUIVOS UTILIZADOS

Não existem.

VEJA TAMBÉM

csplit (comando)

STRIP **Comando - Desenvolvimento**

strip - retira informação simbólica de um arquivo-objeto.

SINTAXE

```
strip [-r] [-x] [-V] arq...
```

DESCRIÇÃO

O comando **strip** retira informação simbólica de arquivos objeto ou de bibliotecas de arquivos-objeto. A finalidade deste comando é a de reduzir o tamanho de arquivos (bibliotecas) objeto, proporcionando assim uma economia de espaço em disco.

OPÇÕES

O comando **strip** sem opções deverá retirar informações de número de linhas e todas as entradas da tabela de símbolos.

-r

Não retira entradas referentes a símbolos estáticos e a símbolos externos, nem informações de relocação.

-x

Não retira entradas referentes a símbolos estáticos e a símbolos externos.

-V

Escreve na saída-padrão de erros a versão do comando **strip** sendo executado.

ARGUMENTOS

arq

Arquivo-objeto ou biblioteca de arquivos-objeto.

DIÁLOGO

Não existe.

CÓDIGOS DE RETORNO

zero - término sem erro.

diferente de zero - término com erro.

OBSERVAÇÃO

Se o arquivo-objeto possui informações de relocação e há informação simbólica a ser removida, **strip** reportará um erro, a menos que a opção **-r** seja utilizada.

ARQUIVOS UTILIZADOS

Não existem.

VEJA TAMBÉM

ar (comando)
cc (comando)
ld (comando)
listver (comando)

Página intencionalmente em branco.

STTY **Comando - Avançado**

stty - modifica atributos do terminal.

SINTAXE

```
stty [-a][-g] [arg...]
```

DESCRIÇÃO

O comando **stty** modifica atributos de E/S de terminal do dispositivo que é a entrada-padrão. Sem argumentos ou opções, são impressos alguns atributos da configuração atual do terminal.

OPÇÕES

-a

Imprime toda a configuração do terminal.

-g

Imprime toda a configuração do terminal de uma forma que possa ser usada como argumento de um outro comando **stty**.

ARGUMENTOS

arg...

Especificam os atributos a serem modificados. O "Manual Administrador do Sistema SOX" (SX03) contém uma descrição detalhada dos modos listados nos primeiros cinco grupos (ver estrutura TERMIO).

Modos de Controle

parenb (-parenb) : ativa (desativa) a geração e detecção de paridade.

parodd (-parodd) : seleciona a paridade ímpar (par).

cs5 cs6 cs7 cs8 : seleciona o tamanho do caractere.

0 : desliga a linha telefônica imediatamente.

número : modifica a velocidade (em bauds) de transmissão / recepção do terminal para o número especificado, se possível.

hupcl (-hupcl) : desliga (não desliga) a conexão com o modem quando o último processo fechar o terminal.

hup (-hup) : igual a hupcl (-hupcl).

cstopb (-cstopb) : o sinal "stop bits" deve ter a duração de transmissão de dois (um) bits para cada caractere.

cread (-cread) : habilita (desabilita) a recepção na linha.

clocal (-clocal) : assume uma linha sem (com) o controle de modem.

Modos de Entrada

ignbrk (-ignbrk) : ignora (não ignora) a ocorrência de "break".

brkint (-brkint) : gera (não gera) o sinal **INTR** na ocorrência de "break".

ignpar (-ignpar) : ignora (não ignora) a ocorrência de erros de paridade.

parmrk (-parmrk) : realiza (não realiza) a marcação dos erros de paridade.

inpck (-inpck) : habilita (desabilita) a verificação de paridade na entrada.

istrip (-istrip) : converte (não converte) os caracteres da entrada para 7 bits.

inlcr (-inlcr) : converte (não converte) **LF** para **CR** na entrada.

igncr (-igncr) : ignora (não ignora) os caracteres **CR** na entrada.

icrnl (-icrnl) : converte (não converte) **CR** para **LF** na entrada.

iuclc (-iuclc) : as letras maiúsculas recebidas são convertidas (não são convertidas) para minúsculas.

ixon (-ixon) : habilita (desabilita) o controle de fluxo através de "xon"/"xoff".
A saída para o terminal é interrompida enviando-se um caractere ASCII **DC3** ("xoff") e iniciada enviando-se o caractere ASCII **DC1** ("xon").

ixany (-ixany) : permite (não permite) que qualquer caractere seja usado como "xon", e não apenas o caractere ASCII **DC1**.

ixoff (-ixoff) : requer que o sistema envie (não envie) "xon"/"xoff" quando a fila de entrada estiver quase vazia/cheia.

Modos de Saída

opost (-opost) : os demais modos de saída são considerados (não são considerados); isso permite otimizar ou não o processamento da saída.

olcuc (-olcuc) : converte (não converte) letras minúsculas destinadas ao terminal em maiúsculas.

onlcr (-onlcr) : converte (não converte) o caractere **LF** destinado ao terminal na seqüência **CR-LF**.

ocrnl (-ocrnl) : converte (não converte) caracteres **CR** destinados ao terminal em **LF**.

onocr (-onocr) : descarta (não descarta) caracteres **CR** escritos logo após outro **CR**, **FF**, **LF** mapeado em **CR-LF** ou com **onlret** ligado, ou seja, com o cursor na coluna zero.

onlret (-onlret) : no terminal o caractere **LF** realiza (não realiza) as funções do **CR**.

ofill (-ofill) : usa caracteres de enchimento para implementação de retardos (usa temporização).

ofdel (-ofdel) : usa o caractere ASCII **DEL** (usa **NUL**) como caractere de enchimento.

cr0 cr1 cr2 cr3 : seleciona o tipo de retardo para o retorno de carro (**CR**).

nl0 nl1 : seleciona o tipo de retardo para avanço de linha (**LF**).

tab0 tab1 tab2 tab3: seleciona o tipo de retardo para a tabulação horizontal (**HT**).

bs0 bs1 : seleciona o tipo de retardo para volta caractere (**BS**).

ff0 ff1 : seleciona o tipo de retardo para alimentação de formulário (**FF**).

vt0 vt1 : seleciona o tipo de retardo para a tabulação vertical (**VT**).

Modos Locais

isig (-isig) : habilita (desabilita) a comparação dos carateres recebidos com os caracteres de controle especiais **INTR** e **QUIT.**

icanon (-icanon) : habilita (desabilita) a entrada em modo canônico.

xcase (-xcase) : transforma (não-transforma) entre caracteres e sequências
x
no modo canônico.

echo (-echo) : ecoa (não-ecoa) todo caractere digitado.

echoe (-echoe) : ecoa (não-ecoa), se eco ativado, o caratere **ERASE** como a sequencia **BS-SP-BS.**

echok (-echok) : ecoa (não-ecoa) o caractere **LF** após o caractere **KILL.**

lfkc (-lfkc) : igual a echok (-echok).

echonl (-echonl) : ecoa (não-ecoa) **LF.**

noflsh (-noflsh) : os caracteres das filas de entrada e saída não são (são) descartados quando ocorrer **INTR** ou **QUIT.**

Atribuições de Controle

caractere-controle c: modifica o caractere-controle para c, onde o caractere-controle pode ser **erase, kill, intr, quit, eol, eof, min** ou **time** (**min** e **time** são usados com **-icanon**).

Se c vier precedido por ^ (circunflexo), significa que o valor a ser tomado é **CTRL** caractere c (por exemplo, **^d** é **CTRL-d**); **^?** é interpretado como **DEL** e **^-** é interpretado como indefinido.

line i : modifica a disciplina de linha para i (0<i<127).

Modos Combinados

evenp ou parity : habilita parenb e cs7.

oddp : habilita parenb, cs7, e parodd.

-parity, -evenp, -oddp : desabilita parenb, e usa cs8.

raw (-raw ou cooked) : habilita (desabilita) E/S no modo "raw".

nl (-nl) : desliga (liga) icrnl, onlcr. O -nl desliga também inlcr, igncr, ocrnl e onlret.

lcase (-lcase) : liga (desliga) xcase, iuclc e olcuc.

LCASE (-LCASE) : igual a lcase (-lcase).

tabs (-tabs ou tab8) : preserva (expande para espaços) tabulações (**HT's**) na saída.

ek : altera os caracteres **ERASE** e **KILL** para os usuais # e @.

sane : coloca todos os modos em valores razoáveis.

DIÁLOGO

Não existe.

CÓDIGOS DE RETORNO

zero - término sem erro.

diferente de zero - término com erro.

OBSERVAÇÕES

. A opção **-g** retorna os modos da estrutura TERMIO em hexadecimal separados por ":".

. Quando um caractere de controle não está definido, aparecerá <**indef**> como seu valor.

ARQUIVOS UTILIZADOS

Não existem.

VEJA TAMBÉM

ioctl (chamada)
Manual Administrador do Sistema SOX (SX03) (estrutura TERMIO)

EXEMPLOS

. Para que os caracteres digitados não apareçam na tela:

```
stty -echo
```

. Para que voltem a sair:

```
stty echo
```

SU **Comando - Avançado**

su - transforma um usuário em superusuário ou em outro usuário.

SINTAXE

```
su [-] [usuário [arg...]]
```

DESCRIÇÃO

O comando **su** permite que um usuário se transforme em outro **usuário**, sem ter que abrir uma nova sessão. O valor default para **usuário** é "root" (isto é, superusuário).

A menos que o usuário já seja superusário, o comando solicita a senha apropriada. Se a senha estiver correta, **su** executa um novo ambiente, com os identificadores de usuário real e efetivo atualizados para os do **usuário** especificado. O novo interpretador de comandos será o programa opcional mencionado na entrada do arquivo de senhas (/etc/passwd), relativa ao **usuário** especificado, ou o default, se nenhum programa for mencionado. Os privilégios originais podem ser restaurados pela digitação de **EOF** (CTRL-D, em geral).

Quaisquer argumentos restantes são passados para o interpretador de comandos.

Se o interpretador de comando especificado no arquivo /etc/passwd for **sh** (veja sh (comando)), então, se o primeiro argumento de **su** for - (hífen), o ambiente será mudado para o ambiente que seria esperado caso o **usuário** tivesse aberto sessão (a Shell é chamada com um arg0 de **-su** ou **-sh**). Caso contrário, o ambiente será mantido, com a possível exceção da variável PATH.

OPÇÕES

Não existem.

ARGUMENTOS

-

Reinicia o ambiente do novo usuário, como se ele tivesse aberto sessão.

arg

Argumentos passados ao interpretador de comandos especificado em /etc/passwd.

usuário

Nome do usuário no qual se deseja transformar.

DIÁLOGO

senha : (pedido de entrada da senha).

CÓDIGOS DE RETORNO

zero - término sem erro.

diferente de zero - término com erro.

ARQUIVOS UTILIZADOS

/etc/passwd
/etc/profile
$HOME/.profile

VEJA TAMBÉM

sh (comando)
Manual de Referência - Linguagem SHELL/SOX (LE01)

Página intencionalmente em branco.

SUM **Comando - Básico**

sum - imprime o "checksum" e contador de blocos de um arquivo.

SINTAXE

```
sum [-r] arq
```

DESCRIÇÃO

Calcula e imprime o "checksum" e o número de blocos de 512 octetos que possui o arquivo passado como argumento.

OPÇÃO

-r

Utiliza um algoritmo alternativo para cálculo do "checksum".

ARGUMENTO

arq

Arquivo a ser processado pelo utilitário.

DIALOGO

Não existe.

CÓDIGOS DE RETORNO

zero - término sem erro.

diferente de zero - término com erro.

OBSERVAÇÃO

Os algoritmos utilizados para calcular o "checksum" serão mantidos uniformes em todas as implementações do SOX, de modo que o mesmo "checksum" será obtido para o mesmo arquivo, independente do hardware e da implementação.

ARQUIVOS UTILIZADOS

Não existem.

SYNC **Comando - Administração**

sync - descarrega buffers do sistema.

SINTAXE

```
sync
```

DESCRIÇÃO

O comando **sync** executa a chamada sync, para forçar a gravação de todos os buffers do sistema que foram modificados, mas que ainda não foram atualizados em disco. Portanto, ele assegura que todas as modificações em arquivos feitas até este ponto estarão salvas.

OPÇÕES

Não existem.

ARGUMENTOS

Não existem.

DIÁLOGO

Não existe.

CÓDIGOS DE RETORNO

zero - término sem erro.

diferente de zero - término com erro.

ARQUIVOS UTILIZADOS

Não existem.

VEJA TAMBÉM

sync (chamada)

SYSDEF **Comando - Administração**

sysdef - lista configuração do sistema.

SINTAXE

```
/etc/sysdef [arqconf]
```

DESCRIÇÃO

O comando **sysdef** analisa o arquivo de configuração fornecido (ou o default, se nenhum foi especificado), reporta eventuais erros e extrai informações de configuração. O arquivo **arqconf** contém os parâmetros de configuração do Núcleo e do Servidor, além da descrição de parâmetros da Máquina Real (relação de PMRs a serem carregados, por exemplo). O arquivo de configuração default é **/conf.sox**.

Se o arquivo de configuração sofreu alguma alteração após a carga inicial do sistema, **sysdef** adverte o usuário de que a descrição fornecida pode não corresponder à efetiva configuração do sistema.

OPÇÕES

Não existem.

ARGUMENTO

arqconf

Arquivo de configuração. Na sua omissão, é examinado o arquivo **/conf.sox.**

DIÁLOGO

Não existe.

CÓDIGOS DE RETORNO

zero	- término sem erro.
diferente de zero	- término com erro.

ARQUIVO UTILIZADO

/conf.sox - arquivo de configuração default.

TABS **Comando - Avançado**

tabs - especifica uma tabulação para um terminal.

SINTAXE

```
tabs [-a] [-a2] [-c] [-c2] [-c3] [-f] [-p]
     [-s] [-u] [-n] [n1, n2, ...]
     [+mmarg_dir] [-Ttipo]
```

DESCRIÇÃO

O comando **tabs** altera a tabulação atual do terminal de acordo com o especificado. A especificação da tabulação pode ser de três tipos: Padrão, Repetitiva e Arbitrária.

Se nenhuma especificação de tabulação é fornecida, o valor -8 é assumido. Opcionalmente pode-se estabelecer uma margem para o terminal.

OPÇÕES

Para especificação de tabulação temos:

a) Tabulação Padrão:

-a

Primeiro formato para ASSEMBLER, sistema IBM/370.
Tabula nas colunas : 1, 10, 16, 36, 72.

-a2

Segundo formato para ASSEMBLER, sistema IBM/370.

Tabula nas colunas: 1, 10, 16, 40, 72.

-c

Formato normal para COBOL.
Tabula nas colunas: 1, 8, 12, 16, 20, 55.

-c2

Formato compactado para COBOL (omite as colunas de 1 a 6).
Tabula nas colunas: 1, 6, 10, 14, 49.

-c3

Formato compactado para COBOL com mais tabulações que c2.
Tabula as colunas: 1, 6, 10, 14, 18, 22, 26, 30, 34, 38, 42, 46, 50, 54, 58, 62, 67.

-f

Formato para FORTRAN.
Tabula nas colunas: 1, 7, 11, 15, 19, 23.

-p

Formato para PL/1.
Tabula nas colunas: 1, 5, 9, 13, 17, 21, 25, 29, 33, 37, 41, 45, 49, 53, 57, 61.

-s

Formato para SNOBOL.
Tabula nas colunas: 1, 10, 55.

-u

Formato para ASSEMBLER do UNIVAC 1100.
Tabula nas colunas: 1, 12, 20, 44.

b) Tabulação Repetitiva:

-n

Inteiro que define tabulações nas colunas 1+n, 1+2*n, etc. Um caso especial é o valor -0, que significa nenhuma tabulação. Se nenhum valor é fornecido, é assumido -8.

c) Tabulação Arbitrária:

n1,n2,...
Inteiros que definem tabulações nas colunas especificadas por n1, n2,...
A numeração das colunas deve ser crescente, e no máximo com 40 números, separados por vírgulas.
A numeração pode ser relativa, por exemplo: 1, +10, +8, 28 que significa tabulação nas colunas 1, 11, 19 e 28.

+m

Especifica a margem direita. Todas as tabulações são deslocadas de n colunas para a direita.

Em muitos terminais para se especificar a margem é necessário fazer **+m0**.

-T

Especifica o tipo do terminal.

ARGUMENTOS

marg_dir

Inteiro que especifica o tamanho da margem, ou seja, o número de colunas a serem deslocadas para a direita.

Se a opção **+m** é fornecida sem um valor para marg_dir, o valor assumido é dez (10).

tipo

Cadeia de caracteres que identifica o terminal. Se omitida, é usado o valor da variável do ambiente TERM. Caso esta variável não esteja definida será assumido o terminal TI200.

DIÁLOGO

Não existe.

CÓDIGOS DE RETORNO

zero - término sem erro.

diferente de zero - término com erro.

OBSERVAÇÕES

. Para o comando **tabs**, a coluna 1 (um) corresponde à coluna mais à esquerda do terminal, mesmo naqueles em que sua numeração é 0 (zero).

. A definição das tabulações e da margem é realizada através da saída-padrão.

. O programa possui uma tabela interna com os comandos de tabulação e margem dos terminais a serem utilizados pelos equipamentos da linha X, não sendo utilizadas as rotinas existentes na biblioteca CURSES, visto que estas não implementam no total todas as opções existentes no comando **tabs**.

. O comando inibe a expansão de tabulações em espaços.

. A tabulação arbitrária, da qual a tabulação padrão é um caso particular, não é aceita pelo terminal TI 200.

ARQUIVOS UTILIZADOS

Não existem.

EXEMPLOS

. tabs -4 -TTI200

Especifica a tabulação nas colunas 5, 9, 13, 17, ... e o terminal usado é TI200.

. tabs -6 +m5

A tabulação será feita nas colunas 11, 17, 23, 29, 35, ...

TAIL **Comando - Básico**

tail - copia porção do final de um arquivo.

SINTAXE

```
tail  [+-[numero][lbc[f]]]  [arq]
```

DESCRIÇÃO

O comando **tail** lista na saída-padrão linhas, caracteres ou blocos do arquivo passado como parâmetro, desde o deslocamento fornecido na opção **+-número** até o final do arquivo. Se nenhum arquivo for especificado é assumida a entrada-padrão

OPÇÕES

Não existem.

ARGUMENTOS

+- numero

Indica o deslocamento em linhas, caracteres ou blocos a partir de onde deve começar a cópia.
+ ou **-** indica se o referencial é o início ou o final do arquivo respectivamente. O valor default é -10.

l

Indica que o argumento número refere-se a linhas (default).

b

Indica que o argumento número refere-se a blocos.

c

Indica que o argumento número refere-se a caracteres.

f

Se o arquivo de entrada não for canal, após terminar a cópia, o comando não termina. Ele "dorme" por um segundo e tenta copiar mais linhas do arquivo de entrada.

arq

Arquivo a ser copiado para a saída padrão.

DIÁLOGO

Não existe.

CÓDIGOS DE RETORNO

zero - término sem erro.

diferente de zero - término com erro.

OBSERVAÇÕES

. Várias anomalias podem acontecer com arquivos especiais tipo caractere.

. A opção **-f** pode ser útil para monitorar o crescimento de um arquivo escrito por um outro processo. Quando se usa essa opção o comando **kill** pode ser empregado para terminar o comando **tail**.

ARQUIVOS UTILIZADOS

Não existem.

EXEMPLO

```
tail -f arq
```

Serão impressas as últimas 10 linhas do arquivo **arq**, mais as linhas que sejam colocadas no final do arquivo **arq**, no espaço de tempo que **tail** tenha iniciado e tenha sido terminado por **kill**.

Página intencionalmente em branco.

TAR **Comando - Avançado**

tar - salva e restaura arquivos.

SINTAXE

```
tar -[rxtuc] [vwfblmo] [arq...]
```

DESCRIÇÃO

O comando **tar** cria conjuntos de arquivos, sua utilização mais comum é para salvar/restaurar arquivos em/de fita magnética. Suas ações são controladas por suas opções. A opção é um - (hífen) seguido de uma cadeia de caracteres contendo uma letra de função, escolhida do conjunto [**rxtuc**], e possivelmente um ou mais modificadores, escolhidos do conjunto [**vwfblmo**].

Outros argumentos são nomes de arquivos (ou de diretórios) especificando que arquivos devem ser salvos ou restaurados. Em qualquer caso, o aparecimento de um nome de diretório refere-se aos arquivos e (recursivamente) aos subdiretórios daquele diretório.

O comando **tar** suporta salvamento/restauração de arquivos em múltiplos volumes. Se o fim do volume é detectado, **tar** solicita a montagem de novo volume para continuar salvando ou restaurando arquivos.

OPÇÕES

Indicadores de função:

-r

Salvamento. Insere os arquivos especificados após o último já inserido.

-x

Restauração. Restaura os arquivos especificados, com o mesmo dono, grupo e data de modificação salvos no conjunto (se possível). Se o arquivo não existir no sistema, o modo salvo no conjunto é restaurado, exceto os indicadores de atualização dos identificadores de usuário e de grupo na execução, que são desligados se o usuário não for superusuário. Se o arquivo já existir no sistema, seu modo não é alterado, a menos do que foi observado anteriormente para os indicadores de atualização dos identificadores de usuário e de grupo na execução. Se nenhum conjunto **arq** for fornecido, todo o conjunto de arquivos é restaurado.

-t

Lista os nomes de todos os arquivos do conjunto.

-u

Salvamento. Insere cada arquivo especificado após o último já inserido se ele não existir no conjunto, ou se existir mas tiver sido modificado após o último salvamento.

-c

Salvamento. Ignora os arquivos já inseridos e começa a inserir os arquivos especificados a partir do início do conjunto.

Modificadores de função:

v

A medida em que trata cada arquivo, imprime o seu nome precedido pela letra de função. Se usada junto com a opção **-t**, imprime também o modo do arquivo, identificador de dono e grupo, tamanho em octetos e a data de modificação. Sem esta opção, o comando não emite mensagem para registrar o sucesso da operação.

w

Antes de tratar cada arquivo pede confirmação do usuário. Se for respondido um dos caracteres [SsYy] a operação com o arquivo é feita. Qualquer outra resposta significa que é para ignorar o arquivo. Esse modificador é ilegal combinado com a opção **-t**.

f

Indica que o argumento seguinte deve ser usado como nome do conjunto, no lugar da fita magnética (/dev/mt). Se o complemento for - indica a entrada padrão para restauração, ou a saída padrão de salvamento. A opção e seu argumento devem ser separados por espaço.

b

Indica que o próximo argumento deve ser usado como fator de blocagem (número de blocos de 512 octetos) que será usado nas operações de entrada e saída. Essa opção é útil para fitas magnéticas usadas no modo "raw", e para maximizar o desempenho das operações de entrada e saída no caso geral. O fator de blocagem pode variar de 1 a 62, desde que válido para o dispositivo em questão. O valor default é 1. O modificador e seu argumento devem ser separados por pelo menos um espaço.

l

Imprime uma mensagem se não for possível resolver todas as associações do arquivo que está sendo salvo. Sem este modificador, o comando não emite mensagem. Válido apenas com as opções **-r**, **-c** e **-u**.

m

Indica que a data de modificação dos arquivos restaurados deve ser aquela em que o comando foi ativado. Válido apenas com a opção **-x**.

o

Indica que os identificadores de dono e de grupo dos arquivos restaurados devem ser os identificadores reais de usuário e de grupo de quem ativou o comando. Válido apenas com a opção **-x**.

ARGUMENTO

arq

Arquivo a ser salvo ou restaurado, podendo ser de qualquer tipo: diretório, regular (strem ou SAX), caractere especial, bloco especial ou FIFO. Pode também ser uma associação.

DIÁLOGO

Não existe.

CÓDIGOS DE RETORNO

zero - término sem erro.

diferente de zero - término com erro.

OBSERVAÇÕES

. Os arquivos são salvos com o mesmo nome com que o usuário os especificou na linha de comando do salvamento. Por exemplo, os comandos para salvar o arquivo /dir/arq

```
$tar -u arq
$tar -u /dir/arq
$tar -u ./arq
```

terão efeitos diferentes na restauração. No primeiro caso, "arq" será restaurado no diretório corrente. No segundo caso, "arq" será sempre restaurado no diretório "dir", independente do diretório-corrente. No terceiro caso, "arq" só vai ser restaurado no diretório corrente, como no primeiro caso, se o comando para restauração especificar exatamente "./arq" ou o diretório ".".

. Na restauração de um conjunto com vários arquivos de nomes idênticos, cada um deles é escrito por cima dos anteriores, de modo que apenas o último será efetivamente restaurado.

. A seguinte linha de comando terá como efeito a cópia de todos os arquivos do diretório "dirorig" para o diretório "dirdest"

```
(cd dirorig; tar -cf - .) |
 (cd dirdest; tar -xf -)
```

. O comando está obedecendo o formato **tar** descrito pelo padrão IEEE 1003.1 (POSIX), suportando também o formato anteriormente usado no UNIX System V.

ARQUIVOS UTILIZADOS

Não existem.

TEE **Comando - Básico**

tee - junta canais e faz cópias da entrada.

SINTAXE

```
tee [-i] [-a] [arq...]
```

DESCRIÇÃO

O comando **tee** transcreve a entrada-padrão para a saída-padrão e para um ou mais arquivos especificados como argumentos.

Se os arquivos não existirem, eles serão criados. Se existirem, a informação é gravada sobre o conteúdo existente.

OPÇÕES

-i

Ignora interrupções.

-a

Faz com que a saída seja anexada ao final dos arquivos, ao invés de escrever sobre a informação existente.

ARGUMENTO

arq

Nome de arquivo que conterá a cópia.

DIÁLOGO

Não existe.

CÓDIGOS DE RETORNO

zero - término sem erro.

diferente de zero - término com erro.

EXEMPLOS

```
. who | tee quem.agora | wc -l
2

. cat quem.agora
teresa     tty1         out  9   14:47
carlos     tty0         out  9   14:45
```

TEST **Comando - Básico**

test - efetua expressões.

SINTAXE

```
test  expressão

[ expressão ]
```

DESCRIÇÃO

O comando **test** avalia **expressão** e retorna zero, se o seu valor for verdadeiro. Caso contrário, retorna um valor diferente de zero.

O comando **test** sem argumentos retorna um valor diferente de zero.

OPÇÕES

Não existem.

ARGUMENTOS

A expressão definida como argumento pode ser construída utilizando os seguintes operadores:

-r arquivo

Verdade, se o arquivo existe, com permissão para leitura.

-w arquivo

Verdade, se o arquivo existe, com permissão para escrita.

-x arquivo

Verdade, se o arquivo existe, com permissão para execução.

-f arquivo

Verdade, se o arquivo existe e é um arquivo regular (stream).

-d arquivo

Verdade, se o arquivo existe e é um diretório.

-c arquivo

Verdade, se o arquivo existe e é um arquivo especial tipo caractere.

-b arquivo

Verdade, se o arquivo existe e é um arquivo especial tipo bloco.

-p arquivo

Verdade, se o arquivo existe e é um arquivo tipo FIFO (canal com nome).

-u arquivo

Verdade, se o arquivo existe e o bit de atualização do identificador de usuário em execução está ligado.

-g arquivo

Verdade, se o arquivo existe e o bit de atualização do identificador de grupo em execução está ligado.

-s arquivo

Verdade, se o arquivo existe e seu tamanho é maior que zero.

-t[n]

Verdade, se o arquivo aberto de descritor "n" (1 por omissão) estiver associado a terminal.

-z s

Verdade, se o tamanho da cadeia "s" for igual a zero.

-n s

Verdade, se o tamanho da cadeia "s" for diferente de zero.

s1 = s2

Verdade, se a cadeia "s1" for igual à cadeia "s2".

s1 != s2

Verdade, se a cadeia "s1" for diferente da cadeia "s2".

s

Verdade, se "s" não for uma cadeia nula.

n1 operador n2

Compara algebricamente os números "n1" e "n2", onde o **operador** pode ser:

-eq (igual);

-ne (diferente);

-gt (maior);

-ge (maior ou igual);

-lt (menor);

-le (menor ou igual).

Qualquer uma das operações acima pode ser combinada com um dos seguintes operadores:

! - operador unário de negação;
-a - operador binário "e";
-o - operador binário "ou";
(expr) - parênteses para agrupar expressões.

DIÁLOGO

Não existe.

CÓDIGOS DE RETORNO

zero - expressão verdadeira.

diferente de zero - expressão falsa.

OBSERVAÇÕES

. Se forem usados caracteres especiais, tais como parênteses dentro do argumento **expressão**, eles devem ser cotados para que a Shell não os interprete.

. Na segunda forma do comando, isto é, a que usa colchetes ([ ]) em vez da palavra **test**, os colchetes devem ser delimitados por brancos.

ARQUIVOS UTILIZADOS

Não existem.

VEJA TAMBÉM

chmod (comando)
find (comando)
mknod (comando)
sh (comando)

Página intencionalmente em branco.

TIME **Comando - Desenvolvimento**

time - imprime estatísticas referentes aos tempos de execução de um comando.

SINTAXE

```
time comando
```

DESCRIÇÃO

O comando **time** faz com que o **comando** passado como argumento seja executado e, após o seu término, informa, na saída-padrão de erros, os tempos gastos na execução deste **comando.**

Estes tempos são: tempo real decorrido durante o comando, tempo de execução de instruções do usuário e tempo de execução de instruções do sistema para a execução do programa do usuário.

Todas as informações são fornecidas em segundos (horas:minutos:segundos, quando necessário).

OPÇÕES

Não existem.

ARGUMENTO

comando

O comando (com seus argumentos) a ser medido.

DIÁLOGO

Não existe.

CÓDIGO DE RETORNO

O mesmo retornado pelo comando que foi medido.

ARQUIVOS UTILIZADOS

Não existem.

OBSERVAÇÃO

Comando não pode ser um "script" para a Shell. Para mensurar o tempo de execução de um arquivo de comandos deve-se fazer **time sh script.**

TOUCH **Comando - Básico**

touch - atualiza tempos de acesso e modificação de arquivos.

SINTAXE

```
touch [-amc] [ddmmhhmm[yy]] arq...
```

DESCRIÇÃO

O comando **touch** atualiza os tempos de acesso e modificação de arquivos. Caso o tempo não seja fornecido será utilizado o corrente. Caso o arquivo mencionado não exista, **touch** o criará, a não ser que a opção **-c** esteja presente.

OPÇÕES

-a

Atualiza somente o tempo de acesso.

-m

Atualiza somente o tempo de modificação.

-c

Não cria o arquivo caso este não exista, não emitindo mensagem de diagnóstico.

ARGUMENTOS

ddmmhhmmyy

Data de modificação ou acesso, no formato:

dd - dia(01-31)

mm - mês(01-12)

hh - hora(01-24)

mm - minuto(01-59)

yy - ano((70-99) ou (00-30)).

Caso a data não seja fornecida, é assumida a data e hora corrente.

arq

Nome do arquivo.

DIÁLOGO

Não existe.

CÓDIGOS DE RETORNO

Número de arquivos que não tiveram suas datas atualizadas, incluindo os arquivos que não existiam e não foram criados (opção -c). Esse número é retornado módulo 101.

OBSERVAÇÕES

- O default da opção é -am.
- Se os dígitos do ano estiverem na faixa (70-99), significa um ano na faixa (1970 a 1999).
- Se os dígitos do ano estiverem na faixa (00-30), significa um ano na faixa (2000 a 2030).

ARQUIVOS UTILIZADOS

Não existem.

VEJA TAMBÉM

umask (comando)

Página intencionalmente em branco.

TR **Comando - Básico**

tr - traduz caracteres.

SINTAXE

```
tr [-cds] [cadeia1 [cadeia2]]
```

DESCRIÇÃO

O comando **tr** copia a entrada-padrão na saída-padrão, substituindo ou removendo os caracteres selecionados. Os caracteres achados na cadeia1 são transformados por correspondência posicional nos caracteres da cadeia2.

OPÇÕES

-c

Complementa o conjunto de caracteres na cadeia1 em relação ao universo de caracteres ASCII, exceto o caractere NUL.

Assim, a cadeia efetiva será composta de todos os caracteres ASCII (em ordem lexicográfica) no intervalo 001 a 377 (octal), menos os fornecidos.

-d

Deleta todos os caracteres de entrada presentes na cadeia1. A cadeia2 não é considerada.

-s

Compacta os caracteres da saída repetidos que estão na cadeia2, em um único caractere.

ARGUMENTOS

cadeia1

Cadeia de caracteres.

cadeia2

Cadeia de caracteres.

DIÁLOGO

Não existe.

CÓDIGOS DE RETORNO

zero - término sem erro.

diferente de zero - término com erro.

OBSERVAÇÕES

. São válidas todas as combinações de opções.

. A notação [a-z] significa a faixa de caracteres de "a" até "z", inclusive, no código ASCII.

. A notação [a*n], representa "n" repetições do caractere "a". No caso de "n" ser zero ou omisso, "n" é interpretado como 300 repetições do caractere "a". Se o número "n" for iniciado por 0 (zero), será considerado octal e, caso contrário, decimal.

. O caractere "\" (contra-barra) seguido por um, dois ou três dígitos octais, envia o caractere ASCII definido por este código.

. O comando **tr** não trata o caractere ASCII nulo na cadeia1 ou cadeia2, e sempre remove este caractere da entrada.

ARQUIVOS UTILIZADOS

Não existem.

EXEMPLOS

. tr "abc" "xy"

Troca "a" por "x", "b" por "y" e "c" por "c" .

. tr "abcde" "[x*2]y[1-3]"

Troca "a" por "x", "b" por "x", "c" por "y", "d" por "1" e "e" por "2".

. tr -cd "aeiouAEIOU" < arq1 | wc -c

Imprime na saída-padrão o número de vogais em arq1.

. tr -cs "[A-Z][a-z]" "[\012*]" <arq1> arq2

Cria em arq2 uma lista das palavras em arq1, uma por linha, onde "palavra" aqui significa uma cadeia máxima de letras.

TRUE **Comando - Básico**

true - retorna condição zero.

SINTAXE

```
true
```

DESCRIÇÃO

O comando **true** nada faz e retorna zero.

OPÇÕES

Não existem.

ARGUMENTOS

Não existem.

DIÁLOGO

Não existe.

CÓDIGO DE RETORNO

zero - término sem erro.

ARQUIVOS UTILIZADOS

Não existem.

VEJA TAMBÉM

false (comando)
sh (comando)

EXEMPLO

```
while true
  do
    comando
  done
```

Loop "infinito" executando comando.

TRUNCX **Comando - Avançado**

truncx - esvazia arquivo(s) regular(es).

SINTAXE

```
truncx arq ...
```

DESCRIÇÃO

O utilitário **truncx** esvazia arquivo(s) regular(es).

OPÇÕES

Não existem.

ARGUMENTO

arq

Nome de caminho de arquivo regular.

DIÁLOGO

Não existe.

CÓDIGOS DE RETORNO

zero - término sem erro.

diferente de zero - término com erro.

ARQUIVOS UTILIZADOS

Não existem.

EXEMPLOS

truncx arqseq arqrel arqind

truncx arq*

TSORT **Comando - Desenvolvimento**

tsort - ordenação topológica.

SINTAXE

```
tsort [arq]
```

DESCRIÇÃO

O comando **tsort** produz na saída-padrão uma lista de itens topologicamente ordenada, conforme uma ordem parcial de itens mencionados no arquivo de entrada. Se o arquivo não é especificado, a entrada-padrão é adotada.

A entrada consiste em pares de itens (cadeia não-vazia) separados por brancos. Diferentes pares de itens indicam a ordenação, sendo o primeiro elemento pré-requisito do segundo. Idênticos pares de itens indicam a presença, mas não a ordem.

OPÇÕES

Não existem.

ARGUMENTOS

arq

Cadeia de caracteres que identifica o arquivo a ser ordenado.

DIÁLOGO

Não existe.

CÓDIGOS DE RETORNO

zero - término sem erro.

diferente de zero - término com erro.

ARQUIVOS UTILIZADOS

/usr/tmp/ nome criado pelo sistema.

VEJA TAMBÉM

lorder (comando)

EXEMPLOS

Dado o arquivo arq, com o seguinte conteúdo:

```
c1 c1
c1 c2
c1 c3
c1 c4
```

```
c3 c10
c3 c6
c3 c7
c3 c9
c4 c10
c4 c7
c4 c8
c10 c11
c10 c7
c11 c12
c13 c13
c13 c14
c13 c4
c14 c15
c14 c2
c14 c3
c15 c5
```

usando o comando

```
. tsort arq
```

```
c1
c13
c4
c8
c14
c2
c3
c10
c6
c7
c9
c11
c12
c15
c5
```

Página intencionalmente em branco.

TTY **Comando - Avançado**

tty - obtém o nome do terminal corrente.

SINTAXE

```
tty [-s]
```

DESCRIÇÃO

O comando **tty** mostra o caminho do terminal do usuário. A opção **-s** inibe qualquer mensagem e produz apenas o código de retorno. Se a entrada-padrão não for um terminal e a opção **-s** não for especificada, uma mensagem será enviada.

OPÇÃO

-s

Inibe a impressão do caminho, ou da mensagem de erro, e apenas retorna o código pertinente.

ARGUMENTOS

Não existem.

DIÁLOGO

Não existe.

CÓDIGOS DE RETORNO

0	- se entrada-padrão for terminal.
1	- qualquer outra condição.
2	- opções inválidas.
maior que 100	- término com erro.

ARQUIVO UTILIZADO

/dev

UMASK **Comando - Básico**

umask - define máscara de modo de criação de arquivo.

SINTAXE

```
umask [nnn]
```

DESCRIÇÃO

O comando **umask** modifica para nnn a máscara de modo de criação de arquivos do usuário. Cada **n** é um dígito octal referente às permissões de leitura, escrita ou execução para o dono, grupo e outros, respectivamente.

Os bits que são iguais a 1 em nnn serão 0 na máscara de modo de todos os arquivos criados após a execução deste comando.

Se nnn for omitido, o valor corrente da máscara é impresso.

OPÇÕES

Não existem.

ARGUMENTO

nnn

Corresponde a três dígitos octais (máscara de criação) que especificam as permissões para leitura, escrita e execução para o dono do arquivo, grupo e outros, respectivamente.

DIÁLOGO

Não existe.

CÓDIGOS DE RETORNO

zero - término sem erro.

diferente de zero - término com erro.

ARQUIVOS UTILIZADOS

Não existem.

VEJA TAMBÉM

chmod (comando)
Manual de Referência da Linguagem Shell/SOX (LE01)

EXEMPLO

umask 022

O exemplo acima remove a permissão de escrita para grupo e outros. O modo normal de criação de arquivos, que era 777, passa a ser doravante 755.

Página intencionalmente em branco.

UMOUNT **Comando - Administração**

umount - desmonta sistemas de arquivo.

SINTAXE

```
/etc/umount especial
```

DESCRIÇÃO

O comando **umount** é utilizado para desmontar sistemas de arquivo. Ele reporta um erro, ao se tentar desmontar o volume-raiz do sistema, volumes que não estejam montados no dispositivo associado ao arquivo especial, ou volumes que estejam ocupados. Um volume está ocupado, se contiver um arquivo aberto, um diretório de trabalho corrente de usuário ou outro volume montado.

Sempre que for realizada uma desmontagem com sucesso, **umount** atualiza a tabela de volumes montados, **/etc/mnttab**, invalidando a entrada correspondente ao volume montado no dispositivo.

OPÇÕES

Não existem.

ARGUMENTO

especial

Arquivo bloco especial ao qual deve estar associado um dispositivo que possua um volume montado.

DIÁLOGO

Não existe.

CÓDIGOS DE RETORNO

zero - término sem erro.

diferente de zero - término com erro.

ARQUIVO UTILIZADO

/etc/mnttab

VEJA TAMBÉM

mount (chamada)

UNAME **Comando - Básico**

uname - imprime o nome do sistema operacional.

SINTAXE

```
uname [-snrvma]
```

DESCRIÇÃO

O comando **uname** escreve o nome do sistema operacional na saída-padrão.

OPÇÕES

-s

Mostra o nome do sistema operacional (default), ou seja, como ele é conhecido na instalação local.

-n

Mostra o nome pelo qual o sistema é conhecido numa rede de comunicações.

-r

Informa a revisão do sistema operacional.

-v

Informa a versão do sistema operacional.

-m

Informa o nome do hardware da máquina.

-a

Mostra todas as informações acima.

ARGUMENTOS

Não existem.

DIÁLOGO

Não existe.

CÓDIGOS DE RETORNO

zero - término sem erro.

diferente de zero - término com erro.

ARQUIVOS UTILIZADOS

Não existem.

VEJA TAMBÉM

uname (chamada)

EXEMPLOS

- uname
  SOX

- uname -a
  SOX local A.00 X10

Página intencionalmente em branco.

UNGET **Comando - Desenvolvimento**

unget - anula a execução anterior do **get** num arquivo SCCS.

SINTAXE

```
unget [-n] [-s] -rsid arq...
```

DESCRIÇÃO

O comando **unget** anula o efeito do comando **get** com a opção **-e**, executado para criar um delta.

Se um diretório é passado como argumento na linha de comando, **unget** procura nesse diretório por todos os arquivos SCCS com permissão de leitura. Caso o caractere "-" (hífen) seja passado, o nome do arquivo SCCS será lido da entrada-padrão.

OPÇÕES

-n

Mantêm o arquivo obtido pelo **get**, que poderia ser normalmente removido do diretório corrente.

-s

Suprime a impressão na saída-padrão do SID correspondente ao delta.

-r

Identifica o delta do arquivo que não será alterado. O uso desta opção é necessário somente se foram solicitadas duas ou mais alterações de versões do mesmo arquivo SCCS, pelo mesmo usuário, através do comando **get**.

ARGUMENTOS

sid

Cadeia com até 19 caracteres que identifica a versão que não será alterada, com o seguinte formato (ver comando **get**):

vvvv.nnnn ou vvvv.nnnn.rrrr.ssss

onde, v, n, r e s são inteiros.

Um erro é informado se o sid especificado é ambíguo, ou omitido na linha de comando.

arq

Cadeia de caracteres que identifica o arquivo tipo SCCS, cujo **get** executado anteriormente será anulado.

DIÁLOGO

Não existe.

CÓDIGOS DE RETORNO

zero - término sem erro.

diferente de zero - término com erro.

ARQUIVOS UTILIZADOS

Não existem.

VEJA TAMBÉM

delta (comando)
get (comando)
sact (comando)

EXEMPLO

. unget -n -s -r1.1.2.1 s.a

O comando **unget** desfaz o efeito do **get** executado com o sid 1.1.2.1 no arquivo "s.a", sem remover o arquivo "a" obtido pelo **get**. Não há impressão do SID na saída-padrão.

Página intencionalmente em branco.

UNIQ **Comando - Desenvolvimento**

uniq - imprime as linhas adjacentes repetidas de um arquivo.

SINTAXE

```
uniq [-cdu] [+n] [-n] [entrada [saida]]
```

DESCRIÇÃO

O comando **uniq** apresenta uma listagem das linhas adjacentes repetidas ou as que não se repetiram, de acordo com a opção escolhida na saída-padrão. As linhas subseqüentes repetidas, são listadas na saída-padrão, porém, com uma só cópia.

OPÇÕES

-c

Força aparecer na saída-padrão uma cópia de cada linha adjacente precedida pelo número de repetições sofridas.

-d

Só aparecem na saída-padrão as linhas que se repetiram, com uma cópia de cada.

-u

Só aparecem na saída-padrão as linhas adjacentes que não se repetiram.

ARGUMENTOS

+n

Faz o interpretador ignorar a existência dos n primeiros caracteres, mesmo existindo campos inteiros antes do n-ésimo caractere em uma linha.

-n

Faz o interpretador ignorar a existência dos n primeiros campos separados por brancos ou caracteres de tabulação em uma linha.

entrada

Arquivo de onde provem as linhas repetidas.

saida

Arquivo para onde vão as linhas repetidas.

DIÁLOGO

Não existe.

CÓDIGOS DE RETORNO

zero - término sem erro.

diferente de zero - término com erro.

OBSERVAÇÕES

. As opções podem ser combinadas entre si e a opção **-c** tem precedência sobre as demais.

. Os argumentos entrada e saída devem ser diferentes. O argumento -n será interpretado antes de +n, caso tenham sido especificados.

ARQUIVOS UTILIZADOS

Não existem.

VEJA TAMBÉM

comm (comando)
sort(comando)

EXEMPLOS

. uniq -ud cadastro

Elimina as linhas repetidas (consecutivas), listando as demais na tela.

. uniq +3 -udc cadastro

As linhas são comparadas, ignorando os três primeiros caracteres, sendo impressas as linhas (completas) e o número de vezes que estas aparecem seguidamente.

UNLINK **Comando - Administração**

unlink - desassocia um arquivo de outro.

SINTAXE

```
/etc/unlink arq
```

DESCRIÇÃO

O comando **unlink** retira uma associação de um arquivo.

O comando **unlink** executa a chamada unlink do sistema, sem tratamento de erro.

OPÇÕES

Não existem.

ARGUMENTO

arq

Nome do arquivo do qual será retirada a associação.

DIÁLOGO

Não existe.

CÓDIGOS DE RETORNO

zero - término sem erro.

diferente de zero - término com erro.

OBSERVAÇÃO

Este comando só pode ser executado pelo superusuário.

ARQUIVOS UTILIZADOS

Não existem.

VEJA TAMBÉM

link (comando)
unlink (chamada)

UNPACK **Comando - Uso Geral**

unpack - descompacta arquivos.

SINTAXE

```
unpack  arq...
```

DESCRIÇÃO

O comando **unpack** descompacta o arquivo que foi compactado anteriormente pelo comando **pack**.

Se o arquivo está compactado, o comando **unpack** descompacta-o, criando uma nova versão. O sufixo **.z** é suprimido do nome do arquivo e se mantêm a permisssão, as datas de criação e modificação e o dono do arquivo compactado.

OPÇOES

Não existem.

ARGUMENTO

arq

Arquivo que se deseja descompactar. Este pode ser passado na forma arq.z ou arq.

DIÁLOGO

Não existe.

CÓDIGOS DE RETORNO

O comando **unpack** retorna um valor correspondente ao número de arquivos que não foi possível descompactar.

ARQUIVOS UTILIZADOS

Não existem.

VEJA TAMBÉM

cat (comando)

EXEMPLO

```
unpack arq1 arq2.z
```

Descompacta arq1.z e arq2.z.

VAL **Comando - Desenvolvimento**

val - verifica as características do conteúdo de um arquivo SCCS especificado.

SINTAXE

```
val [-s] [-mmodulo] [-rsid] [-ytipo] arq...
```

DESCRIÇÃO

O comando **val** verifica se o arquivo especificado é do tipo SCCS, testando as características especificadas nas opções. As opções podem ser fornecidas em qualquer ordem.

Se apenas um - (hífen) for fornecido, a leitura será feita da entrada-padrão até que um "fim-de-arquivo" seja encontrado. Cada linha lida é processada, independentemente, como se fosse uma lista de opções/argumentos.

OPÇÕES

-s

Inibe o envio de mensagens de erro, normalmente geradas na saída-padrão, para qualquer falha detectada no processamento de cada nome de arquivo.

-m

Verifica se o valor da palavra-chave %M% (ver comando **admin**), no arquivo SCCS, coincide com o módulo informado.

-r

Um teste é realizado para determinar se o sid (identificação da versão criada pelo comando **delta**) é ambíguo (por exemplo, -r1 é ambíguo porque fisicamente não existe, porém, implica nos SIDs 1.1, 1.2, etc., os quais podem existir) ou inválido (por exemplo, -r1.0 ou -r1.1.0 são inválidos porque nenhum dos dois casos pode existir como número de versão). Se o sid for válido e não for ambíguo, um teste é feito para determinar se ele realmente existe.

-y

Verifica se o valor da palavra-chave %Y% (ver comando **admin**), no arquivo SCCS, coincide com o tipo informado.

ARGUMENTOS

módulo

Cadeia de caracteres que corresponde ao nome do módulo do arquivo SCCS.

sid

Cadeia com até 19 caracteres que identifica a versão, com o seguinte formato (ver comando **get**):

vvvv.nnnn ou vvvv.nnnn.rrrr.ssss

onde, v, n, r e s são inteiros.

tipo

Cadeia de caracteres que corresponde ao tipo do módulo no arquivo SCCS.

arq

Arquivos tipo SCCS a serem tratados pelo comando.

DIÁLOGO

Não existe.

CÓDIGOS DE RETORNO

O comando **val** gera mensagens de diagnóstico na saída-padrão para cada linha de comando e arquivo processados e retorna um código de 8 bits, ou seja, uma seqüência de bits que ligados possuem a seguinte interpretação (da esquerda para a direita):

bit 0 - argumento arq omitido;

bit 1 - opção desconhecida ou duplicada;

bit 2 - arquivo SCCS defeituoso;

bit 3 - impossibilidade de abertura do arquivo ou arquivo não é SCCS;

bit 4 - SID inválido ou ambíguo;

bit 5 - SID não existe;

bit 6 - %Y% diferente do argumento da opção **-y**;

bit 7 - %M% diferente do argumento da opção **-m**;

ARQUIVOS UTILIZADOS

Não existem.

OBSERVAÇÃO

O comando **val** pode processar dois ou mais arquivos na linha de comando, e, também, pode processar múltiplas linhas de comando (quando lido da entrada-padrão "-"). Nesses casos o código retornado é um **OU** lógico dos códigos gerados para cada linha de comando e arquivo processado.

VEJA TAMBÉM

admin (comando)
delta (comando)
get (comando)
prs (comando)

EXEMPLOS

```
. val -r 1.1 s.arq1 s.arq2
  echo $?
  0
```

Verifica se o sid 1.1 existe nos arquivos s.arq1 e s.arq2. O código de retorno igual a 0 confirma a existência do sid 1.1 nos dois arquivos.

```
. val -r 1.2 s.arq1
  Val: Não existe o sid "1.2" no arquivo "s.arq1"
  echo $?
  4
```

Liga o bit número 5 da condição de retorno, indicando que o sid 1.2 não existe no arquivo s.arq1.

Página intencionalmente em branco.

WAIT **Comando - Básico**

wait - espera o término de um processo.

SINTAXE

```
wait [idp]
```

DESCRIÇÃO

O comando **wait** aguarda o término do processo especificado como argumento e informa seu código de retorno. Sem argumentos, espera por todos os processos-filho ativos e reporta términos anormais.

OPÇÕES

Não existem.

ARGUMENTO

idp

Identificador de processo.

DIÁLOGO

Não existe.

CÓDIGO DE RETORNO

Código de retorno do processo especificado.

OBSERVAÇÃO

idp refere-se a processos síncronos. Caso o idp dado seja de um processo assíncrono, **wait** se comporta como sendo dado sem argumento.

ARQUIVOS UTILIZADOS

Não existem.

VEJA TAMBÉM

wait (chamada)
Manual de Referência da Linguagem SHELL/SOX (LE01)

WALL **Comando - Avançado**

wall - envia mensagem para todos os terminais.

SINTAXE

```
/etc/wall
```

DESCRIÇÃO

O comando **wall** lê uma mensagem de sua entrada-padrão, até que seja enviado um "fim-de-arquivo" (**ALT D**), e imprime nos terminais dos usuários correntes a mensagem precedida por:

Mensagem Geral de <LOGIN-ID> DATA

O remetente deve ser um superusúario para desconhecer qualquer proteção requerida pelos outros usuários e poder enviar-lhes a mensagem (ver comando **mesg**).

OPÇÕES

Não existem.

ARGUMENTOS

Não existem.

DIÁLOGO

Não existe.

CÓDIGOS DE RETORNO

Não existem.

ARQUIVOS UTILIZADOS

/etc/utmp e /dev/tty* (tty* - todos os ttys ativos).

VEJA TAMBEM

mesg (comando)
write (comando)

EXEMPLO

Quando se deseja informar a todos os usuários que o sistema vai ser desligado, a partir da sessão root e do terminal tty0, deve ser digitado:

```
. /etc/wall
  DENTRO DE 5 MINUTOS MANUTENÇÃO DO SISTEMA.
  ALT D
```

No vídeo aparecerá:

```
Mensagem Geral de root tty0 Seg Fev 1 16:21:44 1988
DENTRO DE 5 MINUTOS MANUTENÇÃO DO SISTEMA.
```

Página intencionalmente em branco.

WC **Comando - Básico**

wc - conta linhas, palavras e caracteres.

SINTAXE

```
wc [-lwc] [arq...]
```

DESCRIÇÃO

O comando **wc** conta as linhas, palavras e caracteres de um ou mais arquivos ou da entrada-padrão, esta última se nenhum arquivo for especificado.

Se foi especificado mais de um arquivo, **wc** informa também o total geral de linhas, palavras e caracteres para o grupo de arquivos. Uma palavra corresponde a uma cadeia de caracteres delimitada por espaços, caracteres de tabulação ou de nova linha.

OPÇÕES

Sem opções, o comando **wc** informa o total de linhas, palavras e caracteres. As opções, apresentadas a seguir, podem ser usadas isoladas ou em qualquer combinação. O default é **-lwc**.

-l

Conta as linhas.

-w

Conta as palavras.

-c

Conta os caracteres.

ARGUMENTO

arq

Arquivo a ser processado pelo utilitário.

DIÁLOGO

Não existe.

CÓDIGOS DE RETORNO

zero - término sem erro.

diferente de zero - término com erro.

OBSERVAÇÃO

Quando especificados os arquivos, **wc** imprime seus nomes junto com os contadores.

ARQUIVOS UTILIZADOS

Não existem.

EXEMPLO

```
wc memo1 memo2 memo3
     60       60      240   memo1
     50       50      200   memo2
     40       40      160   memo3
    150      150      600   Total
```

Página intencionalmente em branco.

WHAT **Comando - Desenvolvimento**

what - identifica arquivos SCCS através da ocorrência da cadeia de caracteres @ **(#).**

SINTAXE

what [-s] arq...

DESCRIÇÃO

Pesquisa em cada um dos arquivos fornecidos, a ocorrência da cadeia de caracteres @ (#), e lista o conteúdo do subseqüente até a primeira ocorrência de ", , nova linha, , ou o caractere null. A pesquisa prossegue até que o final do arquivo seja encontrado.

Esta cadeia de caracteres @ (#) é inserida nos arquivos SCCS através da substituição da palavra-chave %Z% na execução do comando **get.** O comando **what** pode ser usado em qualquer arquivo onde a cadeia de caracteres for utilizada.

OPÇÕES

-s

Provoca o término da pesquisa após encontrar a primeira ocorrência da cadeia de caracteres @ (#) em cada arquivo.

ARGUMENTOS

arq

Nome do arquivo-objeto da pesquisa. No caso de mais de um, estes deverão ser separados por espaço.

DIALOGO

Não existe.

CÓDIGOS DE RETORNO

0 - quando encontra qualquer ocorrência do padrão.

1 - em caso contrário.

ARQUIVOS UTILIZADOS

Não existem.

VEJA TAMBÉM

get (comando)

EXEMPLOS

. what -s s.teste

Pesquisa a cadeia de caracteres @(#) no arquivo s.teste e encerra após encontrar a primeira ocorrência.

. what s.teste cadast

Pesquisa a cadeia de caracteres @(#) nos arquivos s.teste e cadast até o final dos arquivos.

Página intencionalmente em branco.

WHO **Comando - Avançado**

who - lista informações sobre usuários ativos no sistema.

SINTAXE

```
who [opções] [arq-who]

who am i

who am I
```

DESCRIÇÃO

O comando **who** pode listar, para cada usuário ativo no sistema, o nome do usuário, linha de terminal, data e hora em que a sessão foi aberta e o identificador de processo (idp) do interpretador de comandos. Na ausência do argumento **arq-who**, essas informações são obtidas do arquivo /etc/utmp.

O comando **who** com os argumentos **am i** ou **am I** identifica o usuário que o chamou.

Exceto para a opção default **-s**, as informações geradas possuem o seguinte formato geral (campos entre colchetes não são sempre impressos):

nome [estado] linha tempo atividade idp [comentário] [cond]

As opções permitem que **ps** liste informações sobre aberturas e fins de sessão, carga do sistema, mudanças do relógio interno do sistema, bem como sobre outros processos (não-associados a usuários) disparados pelo processo **INIT**

OPÇÕES

-a

Processa o arquivo **/etc/utmp** ou o arquivo **arq_who** com todas as opções ligadas.

-b

Mostra a data e a hora da última carga do sistema.

-d

Mostra todos os processos que terminaram e não foram reexecutados por INIT. No campo "cond", para os processos terminados, são mostradas a condição de término e a condição de retorno do processo. O campo "comentário" identifica o processo e é copiado do arquivo **/etc/utmp**, ou **arq_who**.

-H

Imprime as colunas com cabeçalhos.

-l

Lista somente aquelas linhas nas quais o sistema está esperando que o usário abra sessão. No campo nome aparecerá LOGIN. Os outros campos são os mesmos como se o usuário estivesse na linha, exceto o campo "estado", que não existe.

-p

Lista todos os demais processos ativos que foram disparados pelo processo INIT. O campo "nome" é preenchido com o nome do programa encontrado em **/etc/inittab**. Os campos "estado", "linha" e "atividade", não possuem significado.

-q

Mostra os nomes e o total dos usuários com sessões abertas no sistema. Quando esta opção é usada, todas as outras opções são ignoradas.

-r

Mostra o nível de execução corrente do sistema.

-s

Mostra apenas os campos "nome", "linha" e "tempo". Esta é a opção default.

-t

Mostra o último acerto do relógio do sistema.

-T

Semelhante à opção **-u**, porém, o campo "estado" será impresso. O estado indica se qualquer outro usuário pode escrever no terminal. **+** indica que qualquer um pode escrever, caso contrário, **-** será impresso. Se a linha estiver defeituosa, **?** será impresso. O superusuário pode escrever em todas as linhas que tiverem **+** ou **-** no campo "estado".

-u

Lista somente os usuários que possuem sessões abertas.

O formato de saída possui os seguintes campos:

. nome - nome de sessão do usuário.

. linha - nome da linha encontrada no diretório /dev;

. tempo - data e hora que o usuário abriu sessão;

. atividade - número de horas e minutos desde a última atividade ocorrida naquela linha. Um ponto (.) indica que houve, no último minuto, alguma atividade no terminal, sendo este portando, "corrente". Se não houve atividade a mais de 24 horas ou se linha não foi utilizada desde a carga do sistema, esta entrada é marcada como "old".

. idp - identifidor de processo do interpretador de comandos (ou processo de "login");

ARGUMENTO

arq_who

Arquivo examinado pelo **who** para obtenção das informações necessárias. Na sua omissão, é examinado o arquivo /etc/utmp.

DIÁLOGO

Não existe.

CÓDIGOS DE RETORNO

zero - término sem erro.

diferente de zero - término com erro.

ARQUIVOS UTILIZADOS

/etc/utmp
/etc/tty*

Página intencionalmente em branco.

WHODO **Comando - Uso Geral**

whodo - lista os usuários e os processos atuais do sistema.

SINTAXE

/etc/whodo

DESCRIÇÃO

O comando **whodo** informa os usuários que estão utilizando o sistema e os processos a estes associados.

A saída é formada pela combinação das saídas dos utilitários **who** e **ps**, ou seja, informações relativas aos usuários e aos processos. O utilitário fornece também a data e o nome do sistema.

OPÇÕES

Não existem.

ARGUMENTOS

Não existem.

DIÁLOGO

Não existe.

CÓDIGOS DE RETORNO

Não existem.

ARQUIVOS UTILIZADOS

/etc/passwd.

VEJA TAMBÉM

ps (comando)
who (comando)

EXEMPLO

O comando **whodo** fornece uma saída, formada pela combinação das saídas dos comandos **who** e **ps**, conforme exemplo a seguir:

. whodo

```
Qua Nov  2 11:01:15 CST 1988
SOX
tty0     luizusr     08:47
tty1     terezag     10:48
tty2     lys         10:28
tty3     lys         09:49
tty0        3  0:48  -su
tty1      676  0:47  -sh
tty1      764  0:44  /bin/sh
tty1      767  0:41  ps
tty2      440  0:45  -sh
tty3      212  0:46  -sh
tty3      723  0:42  edx
```

A primeira linha fornece a data e a segunda o nome do sistema. As quatro linhas seguintes são obtidas a partir do comando **who** e as restantes do **ps.**

Página intencionalmente em branco.

WRITE **Comando - Avançado**

write - escreve para outro usuário.

SINTAXE

```
write  ident  [terminal]
```

DESCRIÇÃO

O comando **write** é utilizado para estabelecer comunicação via terminal entre dois usuários. O usuário receptor recebe no seu terminal a identificação do usuário que enviará a mensagem, data e a hora, no seguinte formato:

mensagem de <identif. do usuário> (ttyn) data hora

Quando a conexão é completada com sucesso, a campainha do terminal emissor soa duas vezes, indicando que a mensagem digitada está sendo enviada.

Caso o usuário receptor deseje responder a mensagem, deverá então interromper a operação que estiver realizando e digitar

write <identificação do usuário> (ttyn)

quando do recebimento da mensagem inicial.

Tudo que um usuário digita (exceto a tecla escape) é impresso no outro terminal, até que seja enviado um "fim-de-arquivo" (**ALT D**) ou uma "interrupção". Neste ponto o comando **write** escreve "EOT" no outro terminal. O receptor pode, também, interromper outras mensagens que estiverem chegando, executando "**mesg -n**".

Para um usuário que esteja usando mais de um terminal, o argumento terminal deve ser especificado, senão apenas o primeiro a entrar no sistema receberá a mensagem transmitida.

Um usuário pode permitir ou não o envio/recepção de mensagens através do comando **mesg**. Certos comandos não admitem mensagens em seqüência para evitar interferência com suas saídas. Entretanto, se o usuário tem permissões de um usuário privilegiado, as mensagens podem ser forçadas para o terminal inibido para escrita.

Se o caractere "!" é encontrado no início de uma linha, o comando **write** entende que o resto da linha é um comando enviado à **shell** para ser executado no terminal.

OPÇÕES

Não existem.

ARGUMENTOS

ident

Identificador do usuário a quem se destina a mensagem.

terminal

Terminal do usuário receptor (ttyn).

DIÁLOGO

Não existe.

CÓDIGOS DE RETORNO

zero - término sem erro.

diferente de zero - término com erro.

ARQUIVOS UTILIZADOS

/etc/utmp

VEJA TAMBÉM

mesg (comando)
who (comando)

EXEMPLO

. write andrade tty2

A mensagem digitada após este comando será recebida pelo usuário andrade, que está utilizando o terminal tty2. Para finalizar o envio desta mensagem deve ser digitado **ALT D.**

Página intencionalmente em branco.

XARGS **Comando - Desenvolvimento**

xargs - constrói lista(s) de argumentos e exeçuta comando.

SINTAXE

```
xargs [-p] [-t] [-x] [-e cad_fim] [-i cad]
      [-l linha] [-s tamanho]
      [[cmd] [arq inicial... ]]
```

DESCRIÇÃO

O comando **xargs** combina os argumentos especificados com os lidos da entrada-padrão, para executar um comando, uma ou mais vezes.

O número de argumentos e o modo como são combinados, serão determinados pelas opções.

Os argumentos lidos da entrada-padrão devem ser cadeias de caracteres delimitadas por um ou mais caracteres brancos, de tabulação ou nova linha. As linhas vazias serão desprezadas.

Os caracteres branco e de tabulação podem ser considerados como parte de um argumento desde que estejam entre aspas (duplas ou simples), sendo que as aspas delimitadoras são ignoradas.

Outra forma consiste em utilizar o caractere "\" precedendo estes caracteres. Exemplo:

"Cobra Computadores" ou Cobra\Computadores.

A lista de argumentos é construída começando pelos argumentos especificados, seguidos pelos lidos da entrada-padrão.

A opções **-i**, **-l** e **-n** determinam como os argumentos serão utilizados na execução do comando.

Quando nenhuma destas opções for utilizada os argumentos iniciais serão seguidos por argumentos lidos continuamente da entrada-padrão até que o buffer interno esteja cheio, sendo então executado o comando com os argumentos. Este processo se repete até que não existam mais argumentos.

Quando há sobreposição entre opções, a última terá a precedência.

OPÇÕES

Quando nenhuma opção for utilizada, serão executados os comandos com os argumentos iniciais seguidos aos lidos da entrada-padrão.

-p

O usuário é solicitado a confirmar a execução do comando, para cada lista de argumentos. Caso a resposta seja positiva, o comando será executado, se não, será utilizada a próxima lista.

-t

O comando e seus argumentos são listados na saída-padrão de erros no momento de sua execução.

-x

Termina a execução do utilitário se alguma lista de argumentos for maior que o tamanho máximo que deva ser usado. Esta opção deve ser usada quando forem utilizadas as opções **-i** e **-l**. Se nenhuma das opções **-i**, **-l** ou **-n** forem fornecidas, o tamanho total de todos os argumentos é o fornecido na opção **-s**.

-e

O comando **xargs** lê da entrada-padrão até que o fim de arquivo físico ou lógico seja encontrado.

Quando esta opção é utilizada sem argumentos , não existe cadeia que indique fim de arquivo lógico, o final do arquivo é o EOF. Caso a opção **-e** for omitida é assumido - (hífen).

-i

Substitui nos argumentos iniciais, cada ocorrência da cadeia de caracteres (cad) pelo conteúdo de uma linha da entrada-padrão. No máximo cinco argumentos especificados (arq_inicial) podem conter uma ou mais ocorrências da cadeia cad.

Os caracteres branco e de tabulação são ignorados no começo de cada linha. Os argumentos não podem exceder a 255 caracteres.

-l

O comando é executado para um certo número de linhas, contendo argumentos, lidas da entrada-padrão. Um fim de linha é indicado por um caracter de nova linha, porém, se este vier precedido por brancos ou tabulações, a próxima linha será considerada continuação da anterior. O comando é executado para cada linha enquanto o número de linhas lidas da entrada-padrão for menor ou igual a número especificado por linha.

-s

Fixa o comprimento máximo da lista de argumentos.

ARGUMENTOS

cad_fim

Cadeia de caracteres que determina o final lógico do arquivo (lista de argumentos fornecidos pela entrada-padrão).

cad

Cadeia de caracteres que só pode ocorrer no máximo em cinco argumentos especificados (arg_inicial).

linha

Inteiro que especifica o número de linhas com argumentos a serem lidas da entrada-padrão. Se omitido é assumido 1.

tamanho

Inteiro menor ou igual a 470, que especifica o comprimento máximo da lista de argumentos. Se omitido é assumido 470.

cmd

Comando a ser executado. Caso cmd seja omitido é assumido o comando **echo**.

arg_inicial

Argumentos especificados no comando, usados na execução de cmd.

DIÁLOGO

Não existe.

CÓDIGOS DE RETORNO

zero - término sem erro.

diferente de zero - término com erro.

ARQUIVOS UTILIZADOS

Não existem.

VEJA TAMBÉM

echo (comando)

EXEMPLO

Para o arquivo arq cujo conteúdo é:

```
admin
sort
xargs
```

Usando este arquivo como entrada do seguinte comando **xargs**:

```
. cat arg|xagrs -icomando
          echo teste de comando e fim de comando
```

teremos:

```
teste de admin e fim de admin

teste de sort e fim de sort

teste de xargs e fim de xargs
```

YACC **Comando - Desenvolvimento**

yacc - constrói um analisador sintático LR(1), escrito em linguagem "C".

SINTAXE

```
yacc [-dltv] espec
```

DESCRIÇÃO

O comando **yacc** permite a construção de um módulo (analisador sintático) que identifica estruturas sintáticas de um programa e traduz para uma representação interna.

O **yacc** recebe como entrada uma especificação composta de:

- gramática livre do contexto;
- definições;
- declarações;
- trechos de programa escritos na linguagem "C".

A partir desta especificação e de um conjunto de tabelas internas que dirigem a ação de um autômato, o **yacc** constrói um módulo (analisador sintático). Este autômato é implementado pela rotina **yyparse**, que executa um algorítmo de análise sintática (LR1).

A gramática livre do contexto fornecida ao **yacc** pode ser ambígua. Para resolver as ambiguidades existentes, o **yacc** usa regras de precedência e associatividade de operadores opcionalmente fornecidas pelo usuário. Se omitidas são usadas regras internas predefinidas.

A cada regra da gramática o usuário pode associar um trecho de programa escrito na linguagem "C".

O trecho de programa associado a uma regra é denominado "ação correspondente à regra", ou simplesmente "ação". Durante a execução do autômato finito construido pelo **yacc**, cada um destes trechos de programa é executado toda vez que for feita a redução da ação correspondente, onde redução consiste em relacionar um subconjunto da entrada com uma regra predefinida.

A rotina **yyparse** que simula a unidade de controle do autômato finito não faz diretamente a leitura da entrada do programa. Ela recebe "unidades sintáticas" fornecidas por uma outra rotina, denominada **yylex**, que deve ser fornecida pelo usuário.

A rotina **yylex** é chamada por **yyparse** para fazer a leitura do fluxo de entrada do programa, decompondo este fluxo nas unidades sintáticas que são efetivamente tratadas pela rotina **yyparse.** O comando **lex** oferece os recursos necessários a construção automática da rotina **yylex.**

Os símbolos terminais da gramática são unidades sintáticas recebidas pela rotina **yyparse** e fornecidas pela rotina **yylex.** A cada símbolo terminal deve ser associado um número inteiro não negativo diferente de 0 e diferente de 256.

A rotina **yylex** deve apresentar as seguintes características:

- retorna um valor inteiro não negativo (diferente de 0 e 256) a toda leitura de uma unidade sintática;

- retorna um valor menor ou igual a zero quando encontrar fim da entrada (EOF);

Esta rotina pode retornar um valor associado a cada unidade sintática através da variável **yylval.**

Os símbolos terminais são autodefinidos, como palavras reservadas e símbolos especiais (=, ;, etc...). Os símbolos não-terminais são definidos por símbolos terminais e não-terminais, ou seja, êle é recursivo. Exemplo:

comando: **if** expressão **then** comando **else** comando
| **if** expressão **then** comando;

Símbolos terminais: if, then, else e ;.

Símbolos não-terminais: expressão e comando.

Os símbolos não-terminais da gramática são as estruturas de sintaxe reconhecidas por **yyparse.**

Na especificação da gramática livre do contexto tanto os símbolos terminais quanto os não-terminais são representados por "nomes". Os símbolos terminais podem ser também representados por literais da linguagem "C" entre aspas simples.

O nome deve obedecer às seguintes regras:

- começar com letra, "." (ponto) ou "_" (sublinhado);
- conter letra, ".", "_" ou os dígitos de "0" a "9";

O arquivo de saída, **y.tab.c**, deve ser compilado pelo compilador C, gerando um código objeto que deve ser ligado com as rotinas **main**, **yylex** e **yyerror**. Essas rotinas deverão ser fornecidas pelo usuário.

Para facilitar o início da utilização do **yacc** a biblioteca **liby.a** oferece as seguintes rotinas:

main - rotina que chama **yyparse**;

yyerror - rotina que imprime na saída-padrão a mensagem de erro.

Estas rotinas podem ser carregadas através da opção **-ly** do comando **cc**.

TRATAMENTO DE ERROS

Os erros de sintaxe encontrados pelo analisador são tratados da seguinte forma:

1. Ao ser encontrado um erro de sintaxe o analisador se comporta como se tivesse acabado de receber de **yylex** o símbolo terminal **error** (de uso exclusivo do **yacc**, tendo associado a ele o número 256)

2. O analisador procura em seguida, em sua pilha de estados, um estado que contenha uma transição de leitura em **error**.

   Para que este estado exista é necessário que o símbolo terminal **error** tenha sido utilizado em uma ou mais regras da gramática.

3. Se for encontrado tal estado na pilha, todos os estados "acima" dele são removidos (abandonados) da pilha, e o analisador executa a ação de leitura de **error**.

   Após executar a ação de leitura o analisador passa a funcionar em um "modo" especial, denominado "modo de recuperação".

4. Se não for encontrado na pilha do analisador estado com transição em **error** a execução do analisador é abortada, e a rotina **yyparse** retorna um valor não nulo.

Quando o analisador entra em modo de recuperação ele cessa de emitir mensagens de erro de sintaxe até que sejam executadas sucessivamente, de forma correta, três ações de leitura. Desta maneira são evitadas situações de erro decorrentes do primeiro erro.

Se o analisador sintático entrar em estado de "aceitação" a execução da rotina **yyparse** é encerrada, sendo retornado o valor zero.

O usuário pode usar o comando **yyerrok;** em suas ações para terminar o "modo de recuperação", fazendo o analisador retornar ao modo normal ou para forçar o analisador sintático agir como se tivesse havido total recuperação de um erro. Este comando é definido como uma macro no módulo produzido pelo **yacc.**

O comando **yyclearin;**, também definido como macro, permite abandonar o símbolo terminal que estiver na entrada no momento.

TRATAMENTO DE AMBIGÜIDADES

O **yacc** possui duas regras internas para resolver ambigüidades encontradas em uma gramática livre do contexto fornecida a ele como entrada. Estas regras são utilizadas quando o usuário não fornecer definições de precedência e associatividade capazes de orientar o **yacc** na resolução das ambiguidades.

As duas regras internas são:

1. No caso de leitura/redução, a ação escolhida é a de leitura.

2. No caso de redução/redução, a ação escolhida é fazer a redução pela regra que aparecer primeiro na gramática, isto é, pela regra que for apresentada na gramática em linha com número de ordem menor que a linha da regra abandonada.

Quando o usuário fornece definições de precedência/associatividade para os símbolos terminais e regras da gramática, o **yacc** tenta resolver as ambiguidades usando estas definições, antes de aplicar as duas regras internas mostradas acima.

As definições de precedência/associatividade são usadas pelo **yacc** da seguinte forma:

1. No caso de leitura/redução, se tanto o símbolo terminal de entrada quanto a regra da gramática possuirem precedência e associatividade definidos a ação escolhida será:

   . leitura se a precedência do símbolo terminal for maior que a da regra;

   . redução se a precedência da regra for maior que a do símbolo terminal.

Se as precedências forem iguais, a escolha da ação depende da associatividade (que será tam bém a mesma para o símbolo terminal e para a re gra):

. leitura se a associatividade for à direita (**%right**);

. redução se a associatividade for à esquerda (**%left**);

. erro se for não-associativo (**%nonassoc**).

2. No caso de leitura/redução, se ou o símbolo ter minal ou a regra não possui precedên cia/associatividade, as regras internas são aplicadas.

3. No caso de redução/redução, as regras internas são aplicadas.

OPÇÕES

-d

Permite a criação do arquivo **y.tab.h** com as defini ções (#define) dos números associados às unidades sintáticas reconhecidas pelo analisador construido pelo yacc. Cada "#define" associa uma das unidades sintáticas ("token") da gramática a um número in teiro, gerado pelo **yacc** ou fornecido pelo usuário nas especificações de entrada. Este arquivo pode ser referenciado em instruções "#include" coloca das em outros módulos fonte, em especial no módulo que contem a rotina **yylex** usada por **yyparse.**

Com esse "#include" as definições dos números atribuidos às unidades sintáticas podem ser usadas em outros programas-fonte.

-l

Indica que o **yacc** não deve inserir a construção **#line** no módulo criado no arquivo **y.tab.c.** Se omitida esta opção, o **yacc** coloca **#line**" antes de cada trecho de programa "C" fornecido pelo usuário, indicando a sua posição relativa (linha) nas especificações de entrada.

-t

Indica que o código para depuração das especificações de entrada, sempre incluido pelo **yacc** no código "C" gerado em **y.tab.c,** deverá ser compilado. Este código inclui instruções de compilação condicional (#ifdef e #ifndef) que fazem referência ao símbolo **YYDEBUG** tratado pelo pré-processador "C".

Independente da presença da opção **-t,** se o símbolo **YYDEBUG** for diferente de zero no momento da compilação, o código para depuração será compilado.

-v

Faz com que seja criado o arquivo **y.output,** que contem uma cópia da especificação de entrada fornecida ao **yacc,** uma descrição detalhada da máquina LALR(1) construida (com informações sobre conflitos de leitura/ redução e de redução/redução resolvidos pelo mecanismo automático de tratamento de ambigüidades), e informações sobre a execução do comando.

ARGUMENTOS

espec

A especificação de entrada fornecida ao **yacc** pode ser composta de até três partes, separadas pelo sinal "%%":

```
declarações
%%
regras da gramática
%%
programas
```

As sessões de declarações e programas são opcionais. Se a parte de programas não for especificada, o sinal %% que a separa da seção de regras da gramática pode ser omitido.

Na especificação de entrada do **yacc**, os caracteres "espaço", "tab" e "newline" são ignorados. Não podem no entanto, aparecer entre os caracteres que compõem um nome, ou entre os caracteres de símbolos reservados do **yacc.**

Comentários, isto é, texto entre "/*" e "*/", podem ser usados da mesma forma que "espaço", "tab" e "newline" (somente não podem aparecer entre caracteres de nomes e símbolos reservados).

DECLARAÇÕES

Na primeira parte das especificações fornecidas ao **yacc** podem aparecer os seguintes tipos de declaração:

- declarações em linguagem "C", compreendidas entre "%{" e "%}" .

- declarações de elementos da gramática:

  %token
  %start
  %left
  %right
  %nonassoc
  %union
  %type
  %prec

As declarações entre "%{" e "%}" têm como escopo todo o arquivo **y.tab.c** e podem ser utilizadas tanto nas ações (especificadas na seção de regras da gramática) quanto nos programas fornecidos na sessão de programas.

As demais declarações que são descritas a seguir, podem possuir os argumentos nome, num e qualificador, onde:

nome - Representa um nome ou então um literal.

num - Representa um número inteiro positivo.

qualificador - Representa um nome entre "<" e ">".

%token

%token [qualificador] nome [num] [[,] nome

[num]]...

Nomes representando símbolos terminais devem ser declarados na primeira parte; declaração **%token** é uma das formas possíveis.

Após o símbolo reservado **%token** pode aparecer um qualificador usado para associar o token ao membro da união correspondente (ver **%union**).

Os nomes que aparecem depois de **%token** são declarados como representando símbolos terminais. Cada nome não definido que não for declarado dessa forma é considerado um símbolo não-terminal, devendo, portanto, aparecer no lado esquerdo de pelo menos uma regra de gramática.

Depois de cada nome pode ser indicado um número inteiro positivo; este número passa a ser o número associado ao símbolo terminal (unidade sintática). Caso não se associe um número ao nome representando um símbolo terminal, através das declarações, o **yacc** associa automaticamente um número a ele. O **yacc** utiliza números consecutivos, a partir de 257, para os simbolos terminais sem número associado explicitamente.

Os números 0 e 256, são reservados pelo **yacc** para dois símbolos terminais especiais:

0 - corresponde a **$end**, usado para indicar fim da entrada.

256 - corresponde a **error**, usado para sinalizar erro na entrada

O símbolo terminal **error** pode ser usado em regras da gramática, mas não pode aparecer entre as declarações. O símbolo terminal **$end** não pode ser usado pelo usuário, sendo de uso exclusivo do **yacc.**

%start

%start nome

O símbolo não-terminal declarado em **%start** representa a estrutura mais geral descrita pelas regras da gramática, ou seja, é o símbolo inicial da gramática.

Se não for usada a declaração **%start** o **yacc** assume o lado esquerdo da primeira regra como sendo o símbolo inicial da gramática.

%left
%right
%nonassoc

%left [qualificador] nome [num] [[,] nome [num]]...

%right [qualificador] nome [num] [[,] nome [num]]...

%nonassoc [qualificador] nome [num] [[,] nome [num]]...

Estas declarações, além de terem o mesmo efeito da declaração **%token**, isto é, de declarar nomes como símbolos terminais e poder associar a eles um qualificador e números inteiros, determinam a precedência e a associatividade dos símbolos terminais cujos nomes são indicados.

Todos os nomes que aparecerem em uma mesma declaração (**%left, %right** ou **%nonassoc**) têm a mesma prioridade e a mesma associatividade indicada pelo símbolo reservado:

%left indica associatividade "à esquerda";

%right indica associatividade "à direita";

%nonassoc indica não-associatividade.

A ordem em que as declarações aparecem determina o nível de precedência dos símbolos terminais. Os símbolos terminais que aparecerem na primeira declaração têm a prioridade mais baixa; os que aparecerem na última declaração têm a prioridade mais alta.

Um símbolo terminal declarado desta forma não precisa ser declarado também em **%token.**

A prioridade e associatividade de cada símbolo terminal sao usadas para resolver ambigüidades eventualmente existentes na gramática fornecida ao **yacc.**

%union

%union {corpo da declaração de "union"}

Ao construir o analisador sintático a partir das especificações de entrada, o **yacc** assume que existe um valor inteiro associado a cada unidade sintática fornecida pela rotina **yylex**, e que a cada ação associada a uma regra da gramática está também associado um valor inteiro.

O valor associado a cada unidade sintática fornecida por **yylex** a rotina **yyparse**, está na variável **yylval**. Os valores retornados pelas ações estão na "pilha de valores" (ou pilha semântica) do analisador sintático. Estes valores podem ser definidos com outros tipos que não "int", inclusive estruturas.

Neste caso, a pilha de valores do analisador e a variável **yylval** devem ser declaradas como sendo a união ("union") dos tipos desejados, através da declaração **%union**.

O corpo da declaração deve ser apresentado entre "{" e "}" depois do símbolo reservado **%union**.

Ao incluir no módulo produzido as ações especificadas na segunda parte, o **yacc** faz a verificação dos tipos das referências aos elementos da pilha de valores feitas no texto das ações. Quando necessário, qualifica as referências usando os qualificadores declarados na primeira parte (em **%token**, **%left**, **%right**, **%nonassoc** e **%type**).

Opcionalmente, em vez de usar a declaração **%union** para determinar o tipo dos valores associados aos símbolos terminais e à pilha de valores, o usuário pode declarar a união entre os símbolos "%{" e "%}" e usar um typedef para definir a variável **YYSTYPE** para representar essa união.

%type

%type qualificador nome [[, nome]...

A declaração **%type** é usada para associar a cada nome de um símbolo não-terminal o nome de um membro da união dos tipos associados à pilha de valores do analisador (qualificador no nome).

%prec

%prec nome

Por definição, a precedência e a associatividade de uma regra são determinadas pela precedência e associatividade do último símbolo terminal que aparece no "corpo da regra".

Conseqüentemente, uma regra não tem precedência e associatividade no caso de não haver símbolo terminal no "corpo da regra", e no caso do último símbolo terminal não ter precedência e associatividade definidos.

Existem casos em que se faz necessário alterar a definição acima. Para isto se deve utilizar o declarador **%prec** seguido do nome de um símbolo terminal, logo antes da última ação ou do ";" que termina a regra.

Quando é usado o declarador **%prec** a precedência e a associatividade da regra são definidas pelo símbolo terminal cujo nome aparecer depois de **%prec.**

REGRAS DA GRAMÁTICA

A segunda parte da especificação de entrada do **yacc** é composta das regras de uma gramática livre do contexto.

Cada regra apresenta a seguinte sintaxe:

```
nome : corpo da regra ;
```

O "nome" que aparece à esquerda de ":" representa um símbolo não-terminal. O símbolo não-terminal, denominado "lado esquerdo da regra", é definido pelo "corpo da regra" que aparece à direita de ":".

Todo símbolo não-terminal da gramática deve aparecer pelo menos uma vez como lado esquerdo de uma regra.

Se existirem várias regras sucessivas da gramática com o mesmo lado esquerdo, o lado esquerdo pode ser escrito uma única vez e os diversos "corpo da regra" devem ser separados por uma "|" (barra vertical). Neste caso, o ";" somente pode aparecer após o último "corpo da regra".

O "corpo da regra" é constituído de uma seqüência de zero ou mais símbolos terminais e não-terminais.

Opcionalmente, o corpo da regra pode conter também ações, que são trechos de programa escritos em "C" compreendidos entre "{" e "}". Cada ação pode conter um ou mais comandos da linguagem "C".

O analisador sintático construído pelo **yacc** faz com que cada ação seja executada toda vez que a regra onde ela aparece é aplicada, ou seja, quando é feita a redução usando a regra correspondente.

Na maioria dos casos as ações aparecem como último elemento do "corpo da regra" após todos os símbolos terminais e não-terminais. No entanto, uma ação pode aparecer em qualquer lugar no "corpo da regra".

Se o "corpo da regra" contem ações que não são o último elemento do "corpo da regra", o **yacc** decompõe a regra em duas ou mais regras criando novos não símbolos terminais.

Uma ação é um trecho de programa escrito na linguagem "C". O **yacc** permite que nestes trechos de programa sejam utilizadas "pseudo-variáveis" $1, $2, $3, .. que se referem aos valores retornados pelos componentes do lado direito da regra.

A pseudo-variável $1 se refere ao primeiro elemento do lado direito, da esquerda para a direita, $2 se refere ao segundo, e assim por diante.

A pseudo-variável $$ se refere ao valor retornado pela própria regra (valor associado ao lado esquerdo da regra).

Por definição, o valor retornado por uma regra é o valor retornado pelo primeiro elemento do lado direito (se o lado direito for vazio a regra não retorna valor). Se em uma ação for atribuido um valor à pseudo-variável $$, o valor retornado pela regra correspondente será este valor atribuido.

A forma geral de notação de uma pseudo-variável é a seguinte:

```
$[qualificador]num
```

Não são permitidos "espaços" na pseudo-variável.

Um qualificador determina o membro da "união" de valores associados à pilha do analisador sintático que deve ser associado à pseudo-variável.

O uso de qualificador explícito nas pseudo-variáveis é opcional. O **yacc** pode determinar que qualificador usar consultando as declarações de tipo colocadas na primeira parte das especificações.

No entanto, em referências "à esquerda" do contexto da regra (pseudo-variáveis $0, $-1, $-2, ...) o qualificador deve ser especificado. Nestes casos o **yacc** não tem como determinar os tipos.

PROGRAMAS

A sessão de programas deve conter trechos de programas escritos na linguagem "C" O **yacc** simplesmente incorpora estes trechos de programa no arquivo **y.tab.c**,sem dar a eles qualquer tratamento especial.

DIÁLOGO

Não existe.

CÓDIGOS DE RETORNO

zero - término sem erro.

diferente de zero - término com erro.

ARQUIVOS UTILIZADOS

y.output, **y.tab.h** e **y.tab.c**

VEJA TAMBÉM

lex (comando)

EXEMPLO

O seguinte exemplo de especificação para o **yacc** promove a construção de um programa que reconhece expressões aritméticas formadas de constantes numéricas e calcula seus valores.

```
%{
#include <ctype.h>
#include <stdio.h>
#define YYSTYPE double
%}

%token NUMBER
%left  '+' '-'
%left  '*' '/'
%left  UMINUS

%%
lines  :  lines expr '\n' {printf("%g\n",$2);}
       |  lines '\n'
       |
       |  error '\n'      {yyerror("ERRO - ENTRE
                           NOVAMENTE.");
                           yyerrok;}
```

```
        ;

expr    :  expr '+' expr    {$$ = $1 + $3;}
        |  expr '-' expr    {$$ = $1 - $3;}
        |  expr '*' expr    {$$ = $1 * $3;}
        |  expr '/' expr    {$$ = $1 / $3;}
        |  '(' expr ')'     {$$ = $2;}
        |  '-' expr %prec UMINUS {$$ = -$2;}
        |  NUMBER
        ;
%%

yyerror( texto )
char *texto;
{
     printf( "\n%s" , texto );
}

yylex() {

     int prox;
     while( (prox = getchar()) == ' ' );
     if( (prox == '.') || (isdigit(prox)) ){
         ungetc(prox,stdin);
         scanf("%lf", &yylval);
         return NUMBER;
     }
     return prox;
}

main ( argc , argv )
int    argc;
char  *argv[];
{
     int yyparse( void );
     while ( yyparse() );
}
```

Página intencionalmente em branco.

APÊNDICE A

CLASSIFICAÇÃO DOS COMANDOS SOX POR FUNÇÃO

Ambiente de Desenvolvimento de Programas
as, cc, cpp, cxref, lex, m4, make, nl, yacc.

Associação/Desassociação de Arquivos
link, ln, unlink.

Atualização de Manuais
am.

Avaliação de expressões/condições
expr, false, test, true.

Compactação/Descompactação de Arquivos
pack, unpack.

Cancelamento/Suspensão de Processos
kill, killall, sleep.

Comparação de Arquivos/Diretórios
cmp, comm, diff, dircmp, join, paste.

Configuração/Identificação de Terminais
edcar, stty, tabs, tty.

Controle da Execução de Programas
env, nice, nohup, sh/rsh, time, wait, xargs.

Controle de Acesso a Arquivos
chgrp, chmod, chown, umask.

Controle de Acesso ao Sistema
newgrp, passwd, su.

Controle de Programas-Fonte (SCCS)
admin, delta, get, prs, rmdel, sact, unget, val, what.

Cópia/Transferência de Arquivos/Diretórios
copdos, copsel, cp, cpio, cptree, dd, tee, tr.

Criação de Arquivos Especiais/Diretórios
mkdir, mknod.

Criação/Manutenção de Arquivos Seqüenciais, Relativos e Seqüenciais-indexados
creatindx, creatx, delindx, getinfox, manut, recu tra, truncx.

Criação/Manutenção de Bibliotecas
ar, tar.

Desativação do Sistema
sfo, shutdown, sync.

Envio de Mensagens
calendar, echo, mesg, news, wall, write.

Exibição/Atualização de Informações sobre Usuá rio/Arquivos/Sistema
date, devnm, df, du, file, id, logname, ncheck, ps, pwd, sysdef, touch, uname, wc, who, whodo.

Exibição de Arquivos/Diretórios
cat, cut, line, ls, od, pcat, pg, pr, sum, tail, uniq.

Exibição de Calendário
cal.

Exibição de parte de uma Expressão de Caminho
basename, dirname.

Geração/Edição de Textos/Programas
banner, ed, edx, red.

Manipulação de Arquivos-Objeto
ld, listver, lorder, nm, size, strip.

Manipulação de Spool
lpa, lpi, lpl, lpr.

Manutenção do Sistema
bkpcp, clri, fsck, fsdb, formimp, init, iv, mkfs, setmnt.

Montagem/Desmontagem de Volumes
mount, umount.

Ordenação de Arquivos
sort, tsort.

Particionamento de Arquivos
csplit, split.

Procura de Arquivos
find.

Procura de Expressões Regulares/Cadeias em Arquivos
egrep, fgrep, grep.

Remoção de Diretórios
rm, rmdir.

Troca de Diretório
cd, chroot.

Troca de nome/posição de Arquivos e Diretórios
mv, mvdir.

Verificação de Consistência de Arquivos de Senha/Grupo
grpck, pwck.

/dev/tty
Arquivo especial que se refere ao terminal de controle de um processo.
Veja "Manual do Administrador do Sistema SOX" (SX03).

/etc/group
Arquivo regular que contém informações a respeito de grupos.
Veja "Manual do Administrador do Sistema SOX" (SX03).

/etc/passwd
Arquivo regular que contém informações a respeito de usuarios.
veja "Manual do Administrador do Sistema SOX" (SX03).

/etc/utmp
Arquivo regular que contem a relação dos usuários ativos no sistema, bem como a data de carga do sistema, nível de operação corrente, etc.
Veja "Manual do Administrador do Sistema SOX" (SX03).

/etc/wtmp
Arquivo regular que contém um histórico do uso do sistema a partir de sua carga.
Veja "Manual do Administrador do Sistema SOX"(SX03).

ambiente
Conjunto de cadeias de caracteres passado pelas rotinas da família **exec** (ver **exec** (comando)) para um processo. Por convenção, essas cadeias são da forma **variável=valor**, por exemplo PATH=:/bin:/usr/bin.

arquivo de grupo
O mesmo que /etc/group.

arquivo de senhas
O mesmo que /etc/passwd.

arquivo-diretório ou arquivo tipo diretório
Ver diretório.

arquivo especial
Arquivo que está associado a um dispositivo de E/S ou do tipo FIFO.

arquivo FIFO, ou arquivo especial tipo FIFO
Arquivo que implementa um canal de comunicação entre processos, podendo ser referenciado por um caminho.

arquivo regular
Arquivo que não é diretório nem especial.

arquivo regular SAX, ou arquivo SAX
Arquivo regular de uma das organizações estendidas do SOX: seqüencial, relativa e seqüencial-indexada. Ver "Manual de Referência do Sistema de Arquivamento Estendido (SAX)/SOX" (BX01).

arquivo regular "stream", ou arquivo "stream"
Arquivo regular padrão UNIX.

arquivo especial tipo bloco
Arquivo especial que mapeia um dispositivo de E/S. As operações de entrada e saída com tal arquivo passam pelo "buffer pool" do sistema.

arquivo especial tipo a caractere
Arquivo especial que mapeia um dispositivo de E/S. As operações de entrada e saída com tal arquivo, às vezes denominada de modo "cru" ("raw"), são feitas diretamente entre o "controlador" ("driver") do dispositivo e o processo requisitante, sem passar pelo "buffer pool" do sistema.

associação
Diz-se que um arquivo possui uma associação sempre que houver alguma entrada em algum diretório apontando para ele. Um arquivo pode ter uma ou mais associações.

Bloco de Controle de Arquivo (BCA)
Estrutura de dados associada a um arquivo, contendo informações tais como seu tipo, modo, tamanho, datas de último acesso, modificação, ponteiros para os blocos de disco que contêm o arquivo. Reside permanentemente no disco durante a existência do arquivo, e, temporariamente em memória principal, enquanto o arquivo estiver aberto para algum processo.

bit atualiza-identificador-de-grupo-na-execução
Bit do modo de um arquivo que, se ligado no momento da execução deste arquivo, faz com que o identificador efetivo de grupo do processo seja feito igual ao identificador de grupo do arquivo. Como é o identificador efetivo de grupo, e não o identificador real de grupo, que determina permissões de acesso a um arquivo, isto permite que um grupo tenha, temporariamente, as permissões de outro grupo (ver modo do arquivo).

bit atualiza-identificador-de-usuário-na-execução
Bit do modo do arquivo que, se ligado no momento da execução deste arquivo, faz com que o identificador efetivo de usuário do processo seja feito igual ao identificador de dono do arquivo. Como é o identificador efetivo de usuário, e não o identificador real de usuário, que determina permissões de acesso a um arquivo, isso permite que um usuário tenha, temporariamente, as permissões de outro usuário (ver modo do arquivo). Note que, se o identificador de dono do arquivo for zero ("root"), o processo executará com privilégio de superusuário.

bit fecha-na-execução
Bit associado a um arquivo aberto por um processo. Se este bit estiver ligado e o processo executar com sucesso uma chamada **exec** ao Servidor SOX, o arquivo será fechado antes da execução do novo processo. Ver **fcntl** (chamada).

bits de alteração de identificação de usuário e de grupo na execução
Idêntico aos bits atualiza-identificador-de-usuário-na execução e bit atualiza-identificador-de-grupo-na-execução, respectivamente.

bits de permissão de dono
Bits que determinam a permissão de acesso a um arquivo pelo dono do arquivo. Ver modo do arquivo.

bits de permissão de grupo
Bits que determinam a permissão de acesso a um arquivo pelo grupo do arquivo (ver modo do arquivo)

buffer pool
Conjunto de áreas de memória utilizada para armazenamento temporário de informações nas operações de entrada e saída, funcionando como uma extensão dos dispositivos associados.

canal
Mecanismo de comunicação entre processos que se comporta como um arquivo sem nome, onde processos lêem e escrevem, seguindo uma política primeiro-a-entrar-primeiro-a-sair.

caminho (caminho de arquivo)
Cadeia de caracteres utilizada para identificar um arquivo. Corresponde a um mapeamento dentro do diretório hierárquico, que leva a um arquivo de qualquer tipo. Pode ser de dois tipos:

- Caminho absoluto - Começa com um caractere / e o mapeamento é feito a partir do diretório-raiz do processo.

- Caminho relativo - Não começa com o caractere / e o mapeamento é feito a partir do diretório corrente (ver diretório corrente).

data de acesso do arquivo
Data e hora do último acesso ao arquivo.

data de modificação do arquivo
Data e hora da última modificação ocorrida no arquivo.

descritor de arquivo
O mesmo que número lógico.

diretório
Diretórios organizam arquivos em uma estrutura hierárquica (árvore hierárquica, ou sistema de arquivos), no qual os diretórios são os modos intermediários, e as "folhas" são arquivos de qualquer outro tipo.

Um diretório é um arquivo composto de entradas que apontam outros arquivos, incluindo outros diretórios (subdiretórios), que estão diretamente sob ele na hierarquia. Cada entrada do diretório associa um nome de arquivo com identificador de BCA (**ibca**).

Por convenção, todo diretório contém pelo menos duas entradas, . (ponto) e .. (ponto-ponto),que se referem, respectivamente, ao próprio diretório e a seu diretório-pai. O diretório-raiz, que é o nodo de mais alto nível da hierarquia, possui a si próprio como diretório-pai.

diretório-corrente
Diretório usado para facilitar a pesquisa no sistema de arquivamento, pois permite a utilização de caminhos relativos. Todo processo no sistema tem a cada instante um diretório-corrente definido.

diretório-pai
O diretório-pai de um arquivo é um arquivo diretório que contém uma entrada apontando para o arquivo.

diretório-raiz do processo
É a raiz da subárvore do sistema de arquivamento do SOX a qual o processo possui acesso. Pode ser o diretório-raiz do sistema, caso em que o processo possui acesso a todo o sistema de arquivos, ou um diretório qualquer, caso em que o processo só possuirá acesso àquela subárvore. Pode ser alterado dinamicamente para agrupar processos em subárvores específicas do sistema de arquivos. O caminho do diretório-raiz de um processo é / (barra) e seu diretório-pai é **/**.

diretório-raiz do sistema
É a raiz da árvore hierárquica do volume-raiz do sistema.

dispositivo
Arquivo especial (do tipo bloco ou caractere) que está associado a um dispositivo físico de E/S.

dono do arquivo
Identificador associado ao arquivo para controle de permissão de acessos. Inicialmente, é o identificador efetivo de usuário do processo que criou o arquivo, podendo ser alterado dinamicamente.

estado zumbi
Um processo entra no estado zumbi quando, ao terminar, seu pai não está esperando pelo seu término (através da chamada **wait**). Neste caso, suas áreas de programa, dados e pilha são liberadas, mas sua entrada na tabela de processos permanece ocupada.

GMT
É a hora do meridiano de Greenwich, utilizada como padrão para horário absoluto.

grupo do arquivo
Identificador associado ao arquivo para controle de permissão de acessos. Inicialmente, é o identificador efetivo de grupo do processo que criou o arquivo, podendo ser alterado dinamicamente.

grupo de processos
Conjunto de processos relacionados. Estes processos têm em comum o identificador de grupo de processo.

ibca
Identificação de bloco de controle de arquivo (BCA). É o ponteiro utilizado para identificar um arquivo em um diretório.

identificador efetivo de grupo
É a identificação de grupo que um processo assume ao executar um programa. Esta é a identificação utilizada para testar permissões de acesso, relativas a grupo, a recursos do sistema.

identificador efetivo de usuário
É a identificação de usuário que um processo assume ao executar um programa. Esta é a identificação utilizada para testar permissões de acesso, relativas a usuário, a recursos do sistema.

identificador de grupo de processo
É o identificador de processo do líder do grupo de processos ao qual está vinculado o processo em questão.

identificador de grupo de terminal
É uma identificação gerada pelo sistema para controlar processos que estão relacionados a um mesmo terminal.

identificador de pai de processo, ou idp
E o identificador de processo do processo criador do mesmo.

identificador de processo
E uma identificação unívoca gerada pelo sistema para efeito de controle sobre o processo, e que ficará associada ao mesmo desde sua criação até sua extinção.

identificador real de grupo
E uma identificação que todo usuário possui a priori para representar o grupo de usuários ao qual ele pertence.

identificador real do usuário
E uma identificação que todo usuário possui a priori para que o sistema possa diferenciá-los.

líder de grupo de processos
E o processo que controla um grupo de processos. O identificador de grupo de processos dos processos que compõem o grupo de processos é igual ao identificador de processo do líder do grupo.

máscara de criação de arquivo
Máscara utilizada para operar logicamente com o parâmetro **modo** para as chamadas do Servidor SOX **creat, open** e **mknod**, resultando uma máscara de permissão de acessos ao arquivo para dono, grupo e outros, que faz parte do modo do arquivo.

modo do arquivo
E um conjunto de informações associadas ao arquivo contendo o tipo, controles de permissão de acesso e controles de alteração de identificadores na execução do arquivo.

nome de arquivo
Cadeia contendo de 1 a NAME_MAX caracteres, usada para identificar um arquivo (regular, especial ou diretório). Qualquer caractere, exceto **NUL** e **/** (barra), pode fazer parte dessa cadeia. Entretanto, não é aconselhável usar: asterisco, interrogação, abre e fecha colchetes, abre e fecha chaves, maior, menor, aspas simples e dupla, acento grave, barra vertical, circunflexo e abre e fecha parênteses em nomes de arquivo devido ao significado especial que o interpretador de comandos atribui a tais caracteres. Também deve-se evitar hífen, branco, tabulação e caracteres não-visualizáveis.

nome do usuário, ou nome de sessão
É a cadeia de caracteres que é designada a um usuário em uma entrada da tabela de usuários. Segue as mesmas regras de nome de arquivo.

número lógico
Identificação numérica que designa um arquivo aberto pelo usuário. Seu valor vai de 0 a [OPEN_MAX]-1.

octeto corrente do arquivo
Deslocamento (medido em octetos) relativo ao início do arquivo do próximo octeto a ser lido ou gravado. Ao ser criado ou aberto um arquivo regular, o valor do octeto corrente é zero. Uma leitura ou gravação faz com que o valor do octeto corrente seja incrementado do número de octetos lidos ou gravados. Além disto, o valor do octeto corrente pode ser explicitamente modificado pela chamada ao Servidor SOX **lseek**.

processos especiais
Processos responsáveis pela execução de tarefas intrínsecas ao sistema. Os **idps** 0 e 1, pelo menos, são reservados a processos especiais.

sistema-corrente
Sistema local onde o usuário abriu sessão.

stderr
Número lógico associado ao arquivo de saída-padrão de erros. Usualmente, é utilizado o vídeo.

stdin
Número lógico associado ao arquivo de entrada-padrão. Usualmente, é utilizado o teclado.

stdout
Número lógico associado ao arquivo de saída-padrão. Usualmente, é utilizado o vídeo.

superusuário
É uma classe de processos que possui privilégios especiais em operações no sistema. Todo processo que possuir identifidor efetivo de usuário igual a zero é um superusuário.

tabela de processos
Tabela interna do sistema, cujas entradas contêm informações sobre os processos existentes no sistema. Cada entrada corresponde a um processo.

tamanho máximo de arquivos
É um valor associado a cada processo, que limita o tamanho máximo que arquivos podem atingir. Pode ser alterado dinamicamente.

usuário
É a entidade que se utilizará dos serviços do sistema.

volume
É a residência de um sistema de arquivos do SOX, e precisa estar montado sobre um dispositivo de E/S.

volume-raiz
É o volume inicialmente montado no sistema.

VOLUME I

admin . Cria e administra arquivos SCCS.

am . Permite a atualização de manuais.

ar . Editor de bibliotecas.

as . Montador.

banner . Amplia caracteres.

basename . Remove parte inicial de um nome de caminho.

bkpcp . Realiza a cópia de uma unidade de disco.

cal . Imprime calendário.

calendar . Agenda.

cat . Concatena e imprime arquivos.

cc . Compilador C.

cd . Troca o diretório-corrente.

chgrp . Altera grupo de arquivo.

chmod . Altera o modo de arquivo.

chown . Troca o dono de arquivos.

chroot . Troca diretório-raiz para um comando.

clri . Zera BCAs.

cmp	. Compara dois arquivos.
comm	. Seleciona ou rejeita linhas comuns de dois arquivos ordenados.
copdos	. Transporta arquivos entre os sistemas SOX e DOS.
copsel	. Copia arquivos seletivamente.
cp	. Copia arquivos.
cpio	. Cria biblioteca de arquivos e recupera arquivos de bibliotecas criadas por **cpio**.
cpp	. Prê-processador da Linguagem C.
cptree	. Copia um diretório.
creatindx	. Cria índice para arquivo seqüencial--indexado.
creatx	. Cria novo arquivo com organização seqüencial, relativa ou seqüencial--indexada.
csplit	. Particiona arquivos.
cut	. Imprime campos ou colunas de um arquivo.
cxref	. Mostra tabela de referências cruzadas.
date	. Mostra ou atualiza a data e hora.
dd	. Converte e copia um arquivo.

delindx	. Elimina índice de arquivo seqüencial-indexado.
delta	. Cria uma versão de um arquivo SCCS.
devnm	. Retorna nome do dispositivo.
df	. Informa o espaço livre de um volume.
diff	. Compara dois arquivos.
dircmp	. Compara diretórios.
dirname	. Remove parte final de um caminho.
du	. Estima o espaço ocupado por arquivos.
echo	. Ecoa argumentos.
ed	. Editor de texto.
edcar	. Editor de caracteres para terminais baseados no hardware do TI-300.
edx	. Editor de programas.
egrep	. Procura expressões-regulares/ cadeias em arquivos.
env	. Prepara o ambiente para a execução de um comando.
expr	. Efetua expressões aritméticas.
false	. Retorna condição diferente de zero.
ffc	. Formata, verifica a formatação e inicializa fita BKP20.

fgrep	. Procura por cadeias em arquivos.
file	. Determina o tipo do arquivo passado como argumento.
find	. Procura arquivos.
formimp	. Formatador de impressoras.
fsck	. Verifica e repara, interativamente, inconsistências em volumes SOX.
fsdb	. Depurador de volumes.
get	. Obtém uma versão de um arquivo SCCS.
getinfox	. Obtém informações sobre arquivo(s) regular(es).
grep	. Procura expressão-regular/cadeia em arquivos.
grpck	. Consiste arquivo de grupos.
id	. Imprime identificadores e nomes de usuário e de grupo.
init	. Altera o nível de operação do sistema.
iv	. Inicializa e formata volumes de disco.
join	. Junta dois arquivos por campos de igual valor.
kill	. Envia sinal a processo.

killall	. Cancela todos os processos ativos no sistema.
ld	. Ligador.
lex	. Gera programas para análise léxica de texto.
line	. Lê e imprime linha de arquivo.
link	. Associa arquivos.
listver	. Informa versão de arquivos-objeto.
ln	. Cria associação para arquivo.
logname	. Fornece o nome da sessão.
lorder	. Emite uma relação ordenada de nomes de arquivos do tipo objeto ou biblioteca, que estão inter-relacionados.
lpa	. Atua sobre as impressões em curso ou pendentes na impressora lógica.
lpi	. Associa uma impressora lógica a um periférico físico.
lpl	. Mostra a fila de saída do spool.
lpr	. Inclui um arquivo na fila do spool de saída.
ls	. Lista conteúdo do diretório.

VOLUME II

m4	. Processador de macros.
make	. Mantêm, atualiza e gera grupos de programas.
manut	. Verifica/consiste arquivo com organização seqüencial-indexada.
mesg	. Habilita ou não o recebimento de mensagens.
mkdir	. Cria diretórios.
mkfs	. Cria sistemas de arquivos em um volume.
mknod	. Cria arquivos especiais.
mount	. Monta sistemas de arquivos.
mv	. Move arquivos.
mvdir	. Move um diretório.
ncheck	. Lista os nomes dos arquivos associados aos números dos BCAs.
new	. Exibe notícias.
newgrp	. Muda para novo grupo.
nice	. Executa comando com baixa prioridade.
nl	. Filtro de numeração de linhas.

nm	. Imprimea tabela de símbolos de arquivo-objeto.
nohup	. Executa um comando sem interrupção externa.
od	. "Dump" em octal.
pack	. Compacta arquivos.
passwd	. Troca a senha de sessão.
paste	. Combina linhas correspondentes de vários arquivos, ou linhas sub-seqüentes de um arquivo.
pcat	. Lista arquivos compactados.
pg	. Imprime arquivos.
pr	. Imprime arquivos.
prs	. Imprime, na saída-padrão, um arquivo SCCS.
ps	. Informa estados dos processos ativos.
pwck	. Consiste um arquivo de senhas.
pwd	. Imprime o diretório-corrente.
recutra	. Recupera transação.
red	. Editor de texto.
rm	. Remove arquivos ou diretórios.

rmdel	. Remove o delta de arquivo SCCS.
rmdir	. Remove diretórios.
sact	. Lista as versões dos arquivos tipo SCCS editados.
setmnt	. Cria tabela de volumes montados.
sfo	. Finaliza o sistema.
sh/rsh	. Interpretador de comandos padrão/restrito.
shutdown	. Desativa sistema.
size	. Imprime tamanho das seções de arquivos-objeto.
sleep	. Suspende a execução de um processo por um tempo determinado.
sort	. Classifica e/ou intercala arquivos.
split	. Particiona arquivos.
strip	. Retira informação simbólica de um arquivo-objeto.
stty	. Modifica atributos do terminal.
su	. Transforma um usuário em superusuário ou em outro usuário.
sum	. Imprime "checksum" e contador de blocos de um arquivo.

sync	. Descarrega buffers do sistema.
sysdef	. Lista configuração do sistema.
tabs	. Especifica uma tabulação para um terminal.
tail	. Copia porção do final de um arquivo.
tar	. Salva e restaura arquivos.
tee	. Junta canais e faz cópias da entrada.
test	. Efetua expressões.
time	. Imprime estatísticas referentes aos tempos de execução de um comando.
touch	. Atualiza tempos de acesso e modificação de arquivos.
tr	. Traduz caracteres.
true	. Retorna condição zero.
truncx	. Esvazia arquivo(s) regular(es).
tsort	. Ordenação topológica.
tty	. Obtêm o nome do terminal-corrente.
umask	. Define máscara de modo de criação de arquivo.
umount	. Desmonta sistemas de árquivo.
uname	. Imprime o nome do sistema operacional.

unget	. Anula a execução anterior do **get** num arquivo SCCS.
uniq	. Imprime as linhas adjacentes de um arquivo.
unlink	. Desassocia um arquivo de outro.
unpack	. Descompacta arquivos.
val	. Verifica as características do conteúdo de um arquivo SCCS especificado.
wait	. Espera o término de um processo.
wall	. Envia mensagem para todos os terminais.
wc	. Conta linhas, palavras e caracteres.
what	. Identifica arquivos SCCS através da ocorrência cadeia de caracteres.
who	. Lista informações sobre usuários ativos no sistema.
whodo	. Lista os usuários e os processos atuais do sistema.
write	. Escreve para outro usuário.
xargs	. Constrói lista(s) de argumentos e executa comando.
yacc	. Constrói um analisador sintático LR(1), escrito em linguagem C.

ÍNDICE DE FUNÇÕES

VOLUME I

Agenda.	. calendar
Altera grupo de arquivo.	. chgrp
Altera o modo de arquivo.	. chmod
Altera o nível de operação do sistema.	. init
Amplia caracteres.	. banner
Associa arquivos.	. link
Associa uma impressora lógica a um periférico físico.	. lpi
Atua sobre as impressões em curso ou pendentes na impressora lógica.	. lpa
Cancela todos os processos ativos no sistema.	. killall
Compara diretórios.	. dircmp
Compara dois arquivos.	. cmp
Compara dois arquivos.	. diff
Compilador C.	. cc
Concatena e imprime arquivos.	. cat
Consiste arquivo de grupos.	. grpck
Converte e copia um arquivo.	. dd

Copia arquivos.	. cp
Copia arquivos seletivamente.	. copsel
Copia um diretório.	. cptree
Cria associação para arquivo.	. ln
Cria biblioteca de arquivos e recupera arquivos de bibliotecas criadas por **cpio.**	. cpio
Cria e administra arquivos SCCS.	. admin
Cria índice para arquivo seqüencial--indexado.	. creatindx
Cria novo arquivo com organização seqüencial, relativa ou seqüencial--indexada.	. creatx
Cria uma versão de um arquivo SCCS.	. delta
Depurador de volumes.	. fsdb
Determina o tipo do arquivo passado como argumento.	. file
Ecoa argumentos.	. echo
Editor de bibliotecas.	. ar
Editor de caracteres para terminais baseados no hardware do TI-300.	. edcar
Editor de Programas.	. edx

Editor de texto.	. ed
Efetua expressões aritméticas.	. expr
Elimina índice de arquivo seqüencial-indexado.	. delindx
Emite uma relação ordenada de nomes de arquivos do tipo objeto ou biblioteca, que estão inter-relacionados.	. lorder
Envia sinal a processo.	. kill
Estima o espaço ocupado por arquivos.	. du
Formata, verifica a formatação e inicializa fita BKP20.	. ffc
Formatador de impressoras.	. formimp
Fornece o nome da sessão.	. logname
Gera programas para análise léxica de texto.	. lex
Imprime calendário.	. cal
Imprime campos ou colunas de um arquivo.	. cut
Imprime identificadores e nomes de usuário e de grupo.	. id
Inclui um arquivo na fila do spool de saída.	. lpr
Inicializa e formata volumes de disco.	. iv

Informa o espaço livre de um volume.	. df
Informa versão de arquivos-objeto.	. listver
Junta dois arquivos por campos de igual valor.	. join
Lê e imprime linha de arquivo.	. line
Ligador.	. ld
Lista conteúdo do diretório.	. ls
Montador.	. as
Mostra a fila de saída do spool.	. lpl
Mostra ou atualiza a data e hora.	. date
Mostra tabela de referências cruzadas.	. cxref
Obtém informações sobre arquivo(s) regular(es).	. getinfox
Obtém uma versão de um arquivo SCCS.	. get
Particiona arquivos.	. csplit
Permite a atualização de manuais.	. am
Prepara o ambiente para a execução de um comando.	. env
Pré-processador da Linguagem C.	. cpp
Procura arquivos.	. find

Procura expressão-regular/cadeia em arquivos.	. grep
Procura expressões-regulares/ cadeias em arquivos.	. egrep
Procura por cadeias em arquivos.	. fgrep
Realiza a cópia de uma unidade de disco.	. bkpcp
Remove parte final de um caminho.	. dirname
Remove parte inicial de um nome de caminho.	. basename
Retorna condição diferente de zero.	. false
Retorna nome do dispositivo.	. devnm
Seleciona ou rejeita linhas comuns de dois arquivos ordenados.	. comm
Transporta arquivos entre os sistemas SOX e DOS.	. copdos
Troca diretório-raiz para um comando.	. chroot
Troca o diretório-corrente.	. cd
Troca o dono de arquivos.	. chown
Verifica e repara, interativamente, inconsistências em volumes SOX.	. fsck
Zera BCAs.	. clri

VOLUME II

Anula a execução anterior do **get** num arquivo SCCS.	. unget
Atualiza tempos de acesso e modificação de arquivos.	. touch
Classifica e/ou intercala arquivos.	. sort
Combina linhas correspondentes de vários arquivos, ou linhas subseqüentes de um arquivo.	. paste
Compacta arquivos.	. pack
Consiste um arquivo de senhas.	. pwck
Constrói lista(s) de argumentos e executa comando.	. xargs
Constrói um analisador sintático LR(1), escrito em linguagem C.	. yacc
Conta linhas, palavras e caracteres.	. wc
Copia porção do final de um arquivo.	. tail
Cria arquivos especiais.	. mknod
Cria diretórios.	. mkdir
Cria sistemas de arquivos em um volume.	. mkfs
Cria tabela de volumes montados.	. setmnt

Define máscara de modo de criação de arquivo.	. umask
Desassocia um arquivo de outro.	. unlink
Desativa sistema.	. shutdown
Descarrega buffers do sistema.	. sync
Descompacta arquivos.	. unpack
Desmonta sistemas de arquivo.	. umount
"Dump" em octal.	. od
Editor de texto.	. red
Efetua expressões.	. test
Envia mensagem para todos os terminais.	. wall
Escreve para outro usuário.	. write
Especifica uma tabulação para um terminal.	. tabs
Espera o término de um processo.	. wait
Esvazia arquivo(s) regular(es).	. truncx
Executa comando com baixa prioridade.	. nice
Executa um comando sem interrupção externa.	. nohup
Exibe notícias.	. new

Filtro de numeração de linhas.	. nl
Finaliza o sistema.	. sfo
Habilita ou não o recebimento de mensagens.	. mesg
Identifica arquivos SCCS através da ocorrência cadeia de caracteres.	. what
Imprime arquivos.	. pg
Imprime arquivos.	. pr
Imprime as linhas adjacentes de um arquivo.	. uniq
Imprime a tabela de símbolos de arquivo-objeto.	. nm
Imprime "checksum" e contador de blocos de um arquivo.	. sum
Imprime estatísticas referentes aos tempos de execução de um comando.	. time
Imprime, na saída-padrão, um arquivo SCCS.	. prs
Imprime o diretório-corrente.	. pwd
Imprime o nome do sistema operacional.	. uname
Informa estados dos processos ativos.	. ps
Imprime tamanho das seções de arquivos-objeto.	. size

Interpretador de comandos padrão/restrito.	. sh/rsh
Junta canais e faz cópias da entrada.	. tee
Lista arquivos compactados.	. pcat
Lista as versões dos arquivos tipo SCCS editados.	. sact
Lista configuração do sistema.	. sysdef
Lista informações sobre usuários ativos no sistema.	. who
Lista os nomes dos arquivos associados aos números dos BCAs.	. ncheck
Lista os usuários e os processos atuais do sistema.	. whodo
Mantêm, atualiza e gera grupos de programas.	. make
Modifica atributos do terminal.	. stty
Monta sistemas de arquivos.	. mount
Move arquivos	. mv
Move um diretório.	. mvdir
Muda para novo grupo.	. newgrp
Ordenação topológica.	. tsort
Obtêm o nome do terminal-corrente.	. tty

Particiona arquivos.	. split
Processador de macros.	. m4
Recupera transação.	. recutra
Remove arquivos ou diretórios.	. rm
Remove diretórios.	. rmdir
Remove o delta de arquivo SCCS.	. rmdel
Retira informação simbólica de um arquivo-objeto.	. strip
Retorna condição zero.	. true
Salva e restaura arquivos.	. tar
Suspende a execução de um processo por um tempo determinado.	. sleep
Traduz caracteres.	. tr
Transforma um usuário em superusuário ou em outro usuário.	. su
Troca a senha de sessão.	. passwd
Verifica as características do conteúdo de um arquivo SCCS especificado.	. val
Verifica/consiste arquivo com organização seqüencial-indexada.	. manut

COMPUTADORES E SISTEMAS BRASILEIROS S.A

CARTÃO DE CADASTRO E COMENTÁRIOS DO USUÁRIO

Manual de Referência Comandos SOX SX0188-03.0

Prezado leitor:

Suas críticas, comentários e sugestões são valiosos instrumentos para que a COBRA possa, cada vez mais, aprimorar a documentação de seus produtos. Solicitamos, portanto, sua colaboração em preencher este cartão, colocando-o em qualquer caixa de correio (porte pago). Obrigado.

AVALIAÇÃO DO MANUAL

Nos itens que se seguem, indique o conceito aplicável, comentando se assim o desejar:

CONCEITOS: • Excelente (E) • Bom (B) • Regular (R) • Precisa melhorar (P) • Insuficiente/Inadequado (I)

ITEM	CONCEITO
APRESENTAÇÃO DO MANUAL (Encadernação, acabamento, formato)	
QUALIDADE GRÁFICA (Tipo de impressão, diagramação, papel utilizado)	
CLAREZA DAS INFORMAÇÕES (Linguagem empregada, facilidade de compreensão do texto)	
ABORDAGEM DOS ASSUNTOS (Organização, precisão e objetividade)	

ITEM	CONCEITO
DESENHOS, ILUSTRAÇÕES E EXEMPLOS (Suficiência, adequabilidade e clareza)	
UTILIDADE (Adequação à utilização pretendida, facilidade de consulta)	
EDITORAÇÃO E REVISÃO (Existência de falhas de impressão, paginação, erros ortográficos ou gramaticais)	
QUALIDADE GERAL DO MANUAL (Avaliação global do usuário)	

COMENTÁRIOS E SUGESTÕES

IDENTIFICAÇÃO DO USUÁRIO

Nome:	Profissão:
Empresa:	Cargo:
Endereço p/correspondência:	CEP:
	Tel. :

Av. Comandante Guaranys, 447 - 22775 - Rio de Janeiro - RJ
(021) 342-9393 - Telex (021) 22420

COLE ESTA ABA NO VERSO DO CARTÃO

ISR-52-850/87
UP APT PRES. VARGAS
DR-RJ

CARTA RESPOSTA COMERCIAL

NÃO É NECESSÁRIO SELAR

O SELO SERÁ PAGO POR

COBRA COMPUTADORES E SISTEMAS BRASILEIROS S/A

(DIV. ASSIST. SOFTWARE - DAS)

20299 RIO DE JANEIRO - RJ

Remetente:

Endereço:

CEP

COLE AQUI A ARA DO CARTÃO

cobra

Computadores e Sistemas Brasileiros S.A.