Y- matches only whole words only
w- displays lines that do not match the pattern
v- displays line numbers
n- displays list of lines containing pattern
l- displays list of lines containing pattern
- do not list file names if many files are specified
-

pattern : regular expression

display lines from files that match the pattern

*{global pattern} {filename}

${filtering patterns: grep, fgrep, grep}

Ch. 7 Utilities
preceding character; ex. (ab | b*a)
- `diff` - The `diff` command compares two files and outputs a description of their differences.
- `diff -1` - The `diff` command suppresses all output for mismatched bytes.
- `diff -s` - The `diff` command displays the line number and byte offset of all mismatches.
- `diff -1 -s` - The `diff` command displays the line number and byte offset of all mismatches, where comparison begins.

Optional values: `offset1` and `offset2` are the offsets into the files.
- If one file is a prefix of the other, EOF message is displayed.
- There is a mismatch.

`comp -l` - The `comp` command compares two files for equality; reports the first byte where the first file differs from the second file.
Recursively.

If tilde list contains directories, its contents are appended/extraced.

\(-v|--verbose\)

\(-f|--option enables you to give a tar file name (default is /dev/rmt0)

\(-a|--archived\)

\(-n|--option appends only tildes that are more recent than those already

\(-u|--option unconditional appends tildes to tar formatted file

\(-t|--option generates a table of contents

\(-x|--option extracts files from the tar-formatted file

\(-c|--option creates the tar-formatted file

\(-r|--create a tar-formatted file from the tilde list

\(-y|--create y tape archive (tape archiver) file from the tilde list

Archiving:
true if the group of the title is equal to the group of the owner of the title
(true if the type of the title is user

(block, c=char) (p=block, c=char) (c=char) (p=block)

true if the octal description of the title’s permission
true if the octal name matches pattern

-----------
Expression

Value/Action

the titles etc.
allows you to perform certain actions such as deleting
pathlist and descendant from there on, but also
Find titles starting at

Find pathlist expression -
exprt -o exprt2 short circuit or
exprt -a [exprt2 short circuit and

expression

expression negation of expression

display the current title's attributes and returns true

prunes out the name of the current title and returns true

replaced by the title name currently matched

If it is specified as a command line argument it is

command must be terminated by \'

exec command true if the exit code = 0 from executing the command.

title attributes have been modified

modified within count days or any of its
count title attributes have been modified within
count title days

count title if the title has been accessed within

count title if the contents of the title have been

---------- ----------

Expression Value/Action
ls and then remove all titles that end with .bak

\{
} \} \{ \}
$ find . -name '*.bak' -ls -exec rm \\{

It's titles modified in the last 14 days

find . -mtime 14 -ls

searches for title x.c in the entire title system

cfg/junk/cfg

cfg/junk/cfg/cfg

cfg/junk

cfg/junk

* find cfg -name *

$
30 15 1 * 1 mail users Jan meeting at 3pm
* * * * echo One Minute passed
0 8 * 1 echo Happy Monday Morning
$ cat cron.
cron
ex.

following time format must be created:
minute hour day month weekday

The cron tab utility allows you to schedule a series of jobs to be executed on a periodic basis. A title with the
command (any Unix command)

- Scheduled commands: cron tab and at

- [username] cron tab

- [username] cron tab

- [username] cron tab
unregister crontab file
$ crondb -r

30 15 1 * 1 mail users % Jan meeting at 3pm
   echo One Minute Passed
   echo Happy Monday Morning
0 8 * 1 echo %
($ crontab -- $ID: crondb.c'v 2.13 1994/01/17 03:20:37 vixie Exp $)
($ crontab.cron installed on Sat Jun 26 23:33:35 1999)
DO NOT EDIT THIS FILE - edit the master and reinstall.
$ crondb -l
$ crontab crondb.cron
$
(By minutes/hours/days/weeks/months/years)

Stated time may be incremented by an increment (number followed by an increment)

Keyword now can be used in place of time sequence

date is spelled out using first 3 letters of day and/or month

time is specified as HH or HHMM followed by an optional AM/PM

-1 option: list entries in queue
-1 option: remove entry from queue
-m option: send mail
-s option: use Bourne shell
-c option: use C-shell

+ [jobid] at
+ [jobid] at

at -cm time [date,] [year][+increment] [script]

at command allows you to schedule one-time commands/scripts
at now + 2 minutes at . cash

date > /dev/tty

# /bin/csh

You may program the script to reschedule itself as follows:

# echo at done > /dev/tty

# /bin/csh

$ cat at . cash

#
(could have other separators specified by \$F option)

tabs/spaces (could have other separators specified by \$F option)

awk reads a line; it breaks it into fields separated by

The conditions and actions are specified in an awk program.

action on all lines that match a particular condition.

awk: utility that scans one or more files and performs an
- list of statements
  - exit (skips the rest of the current line)
  - next (skips remaining patterns on current line of input)
  - printf; printf;
  - assignment statement
  - if-else; while; for; break; continue
  - action is one of the following kinds of C-like statements

and/or regular expressions

- an expression involving logical operators, relational operators,
  - special tokens BEGIN or END

where condition is one of the following:

[ [ condition [ \ action ] ] ]

awk programs has one or more commands of the form:

-
To execute the above program, use

```awk
% awk -f P2.awk /etc/passwd
```

**FILENAME**: built-in variable for name of file being processed

```awk
BEGIN { BEGIN "Start of FILENAME" }
END { print "End of FILENAME" }
{ END }
BEGIN { BEGIN "Start of FILENAME" }
END { print "End of FILENAME" }
{ "" BEGIN "Start of FILENAME" }
END { print "End of FILENAME" }
```

- BEGIN condition is triggered before the first line is read and
- the END condition is triggered after the last line is read.

Prints the number of fields and the first field in the `/etc/passwd` file

```awk
% awk -f P2.awk /etc/passwd
```

- `FILENAME` refers to name of file
- `NF` refers to number of fields
- `0` refers to entire line
- access to individual fields: `$1`, `$2`, `$3`, ...,$n refer to fields 1 through n

```awk
{ END }
BEGIN { BEGIN "Start of FILENAME" }
```

Prints the number of fields and the first field in the `/etc/passwd` file

```awk
{ BEGIN }
```

Prints the number of fields and the first field in the `/etc/passwd` file

```awk
{ END }
```

Prints the number of fields and the first field in the `/etc/passwd` file

```awk
{ BEGIN }
```

Prints the number of fields and the first field in the `/etc/passwd` file

```awk
{ END }
```

Prints the number of fields and the first field in the `/etc/passwd` file

```awk
{ BEGIN }
```

Prints the number of fields and the first field in the `/etc/passwd` file

```awk
{ END }
```

Prints the number of fields and the first field in the `/etc/passwd` file

```awk
{ BEGIN }
```

Prints the number of fields and the first field in the `/etc/passwd` file

```awk
{ END }
```

Prints the number of fields and the first field in the `/etc/passwd` file

```awk
{ BEGIN }
```

Prints the number of fields and the first field in the `/etc/passwd` file

```awk
{ END }
```

Prints the number of fields and the first field in the `/etc/passwd` file

```awk
{ BEGIN }
```

Prints the number of fields and the first field in the `/etc/passwd` file

```awk
{ END }
```

Prints the number of fields and the first field in the `/etc/passwd` file

```awk
{ BEGIN }
```

Prints the number of fields and the first field in the `/etc/passwd` file

```awk
{ END }
```

Prints the number of fields and the first field in the `/etc/passwd` file

```awk
{ BEGIN }
```

Prints the number of fields and the first field in the `/etc/passwd` file

```awk
{ END }
```

Prints the number of fields and the first field in the `/etc/passwd` file

```awk
{ BEGIN }
```

Prints the number of fields and the first field in the `/etc/passwd` file

```awk
{ END }
```

Prints the number of fields and the first field in the `/etc/passwd` file

```awk
{ BEGIN }
```

Prints the number of fields and the first field in the `/etc/passwd` file

```awk
{ END }
```

Prints the number of fields and the first field in the `/etc/passwd` file

```awk
{ BEGIN }
```

Prints the number of fields and the first field in the `/etc/passwd` file

```awk
{ END }
```

Prints the number of fields and the first field in the `/etc/passwd` file

```awk
{ BEGIN }
```

Prints the number of fields and the first field in the `/etc/passwd` file

```awk
{ END }
```

Prints the number of fields and the first field in the `/etc/passwd` file

```awk
{ BEGIN }
```

Prints the number of fields and the first field in the `/etc/passwd` file

```awk
{ END }
```

Prints the number of fields and the first field in the `/etc/passwd` file

```awk
{ BEGIN }
```

Prints the number of fields and the first field in the `/etc/passwd` file

```awk
{ END }
```

Prints the number of fields and the first field in the `/etc/passwd` file

```awk
{ BEGIN }
```

Prints the number of fields and the first field in the `/etc/passwd` file

```awk
{ END }
```

Prints the number of fields and the first field in the `/etc/passwd` file

```awk
{ BEGIN }
```

Prints the number of fields and the first field in the `/etc/passwd` file

```awk
{ END }
```

Prints the number of fields and the first field in the `/etc/passwd` file

```awk
{ BEGIN }
```

Prints the number of fields and the first field in the `/etc/passwd` file

```awk
{ END }
```

Prints the number of fields and the first field in the `/etc/passwd` file

```awk
{ BEGIN }
```

Prints the number of fields and the first field in the `/etc/passwd` file

```awk
{ END }
```

Prints the number of fields and the first field in the `/etc/passwd` file

```awk
{ BEGIN }
```

Prints the number of fields and the first field in the `/etc/passwd` file

```awk
{ END }
```

Prints the number of fields and the first field in the `/etc/passwd` file

```awk
{ BEGIN }
```

Prints the number of fields and the first field in the `/etc/passwd` file

```awk
{ END }
```

Prints the number of fields and the first field in the `/etc/passwd` file

```awk
{ BEGIN }
```

Prints the number of fields and the first field in the `/etc/passwd` file

```awk
{ END }
```

Prints the number of fields and the first field in the `/etc/passwd` file

```awk
{ BEGIN }
```

Prints the number of fields and the first field in the `/etc/passwd` file

```awk
{ END }
```

Prints the number of fields and the first field in the `/etc/passwd` file

```awk
{ BEGIN }
```

Prints the number of fields and the first field in the `/etc/passwd` file

```awk
{ END }
```

Prints the number of fields and the first field in the `/etc/passwd` file

```awk
{ BEGIN }
```

Prints the number of fields and the first field in the `/etc/passwd` file

```awk
{ END }
```

Prints the number of fields and the first field in the `/etc/passwd` file

```awk
{ BEGIN }
```

Prints the number of fields and the first field in the `/etc/passwd` file

```awk
{ END }
```

Prints the number of fields and the first field in the `/etc/passwd` file

```awk
{ BEGIN }
```

Prints the number of fields and the first field in the `/etc/passwd` file

```awk
{ END }
```

Prints the number of fields and the first field in the `/etc/passwd` file

```awk
{ BEGIN }
```

Prints the number of fields and the first field in the `/etc/passwd` file

```awk
{ END }
```

Prints the number of fields and the first field in the `/etc/passwd` file

```awk
{ BEGIN }
```

Prints the number of fields and the first field in the `/etc/passwd` file

```awk
{ END }
```

Prints the number of fields and the first field in the `/etc/passwd` file

```awk
{ BEGIN }
```

Prints the number of fields and the first field in the `/etc/passwd` file

```awk
{ END }
```

Prints the number of fields and the first field in the `/etc/passwd` file

```awk
{ BEGIN }
```

Prints the number of fields and the first field in the `/etc/passwd` file

```awk
{ END }
```

Prints the number of fields and the first field in the `/etc/passwd` file

```awk
{ BEGIN }
```

Prints the number of fields and the first field in the `/etc/passwd` file

```awk
{ END }
```

Prints the number of fields and the first field in the `/etc/passwd` file

```awk
{ BEGIN }
```

Prints the number of fields and the first field in the `/etc/passwd` file

```awk
{ END }
```

Prints the number of fields and the first field in the `/etc/passwd` file

```awk
{ BEGIN }
```
END \{ \texttt{printf} \%s
\"\texttt{line \%d:}\%s\n\"\ NR, \$0
\} \}

BEGIN \{ \texttt{printf} \%s\n\"\texttt{S\_c\_a\_n\_a\_n\_i\_g \_f\_i\_t\_e}\"\}

\texttt{p4\_awk}

- \texttt{vp\_variables}

\texttt{built-in \texttt{variables}} \texttt{NR} \texttt{contains the current line number}

\{ \texttt{NR} > 1 \&\& \texttt{NR} > 4 \} \texttt{printf} \%s
\texttt{NR, \$1, \$6, \$NF}

\texttt{p3\_awk}

- \texttt{operators}
```bash
awk -F: 'print substr($1,1,2), /etc/passwd

- awk -F: /root:/nobody /etc/passwd

- Condition ranges: two expressions separated by comma
```
Is -l will display the contents of the symbolic link.
Use ls -ll to view details (which title it refers to) of the link.

Another title system: Hard links are restricted to one title system
Symbolic/soft links can be created from one title system into

% ln -s original newlink
Soft (symbolic) links:

If newlink is a directory then links are make within the directory

When both labels are removed,
Both labels will refer to the same file, file will be deleted only
called newlink;
This command creates a hard link to the original title

% ln original newlink
Hard links:
If username is not specified, root is assumed.

```
% su [-] username
```

- Substitute user
Key could be any string:

```
% crypt key > sample.crypt > sample.txt (to uncrypt)

% crypt key > sample.crypt > sample.crypt (to crypt)

% gunzip filename
% gzip filename

% compress filename
% uncompress filename

- compress/uncompress
```
sed does not modify the input file; it just writes modified file to standard output.

-sed (stream editor) scans one or more text files and performs an editing action on all the lines that match a condition.

- actions and conditions may be stored in a file or may be specified on the command line within single quotes.

- sed commands begin with an address or an address-range or a Regular expression.

- actions and conditions may be stored in a file or may be specified on the command line within single quotes.
read commands up to the matching } as a group

set Label for Q and T

set commands

do see cmd if line is not selected

print current input line number

del character from str2 (no ranges allowed)

replace each character from str1 with corresponding

while line(s) to file

lost branches to label if substitution made to current line

if = i
toggle while loop

if = f, print

f, replace all occurrences

substitute part2 for part1

read line, copy contents to stdout

read file (or scripts)

print line

insert following text before next output

del all lines

change lines to following text (as in a)
will add two lines in the beginning of the title
% sed -i 'sed_title > title.new'

Etc
\abcd
11
cat sed
%
deltes all lines containing 'a'
% sed 'd title > title.new

removes all leading spaces from each line of the title
% sed 's/ title > title.new

indents each line in the title by 2 spaces
% sed 's/ title > title.new

Examples:
  - substitute text:
% cat sed3
1c
Line 1 is censored
2c
Line 2 is censored
3c
Line 3 is censored

Multiple commands; individual lines are replaced will replace lines 1–3 by lines 1–3 are censored

1,3c
Lines 1–3 are censored
multiple commands (e option is optional) means script on command line

```
% sed -e "s/\$\(.*\)/$\1/" << /dev/stdin
```

appends title at end of title file

```
% sed 's/.*$/\$TITLE\$< \
```
to one occurrence
s option causes every repeated character in string 1 to be condensed
input
-d option causes every character in string 1 to be deleted from std.
-strin 1 now is in string 2 (!)
-c option causes string 1 to be complemented (every character not in
string 2 is less, the last character is repeated.
set string 1 to the corresponding character in string 2; if length of
this command maps all characters in std. input from character

% tr -cs string 1 string 2

-tr utility (translate)
causes all a, b, c to be deleted from titl1
% tr -d a-c > titl1 < titl2

causes all non-alphabet \127ic characters to be replaced by ascii 12 (n1)
% tr -c a-z'\127 > titl1 < titl2

causes every non-a character to be replaced by \x (including n1)
% tr -c a x > titl1 < titl2

causes lower-case to upper-case conversion
% tr a-z A-Z > titl1 < titl2

Examples:
# concatenation
print "A", "B"
print "AB", "BBCC", "CDD", "\n";
print 1, 2, 3.15, "\n";
print 1, 2, 3.

# strings
# is available.

In addition to standard arithmetic operators, a range operator

```perl
$s = 3;
$s = $s + 2;
print "$s
";
```

Variables (untyped) always begins with a $.

```perl
print "hello world\n";
```

### Printing text:

- Perl takes its syntax and features from both Shell scripts; C programming
- Perl
```c
||
-- --
++

/ *, *
+

- MathematiCal And LogiCal Operators:

011 print @arr + @arr; # print 20
print @arr, "\n"; # will print 12345
print @arr[3][1] = 1.15);
@arr = (1, 2, 3, 4, 5);

Array Index Begins at 0
Arrays: Dynamically Allocated; Array Names Begin With @ Symbol
```
{ "\n", 'i
' = t
" print
    } (I'
R' foreach i
} (I
A
B
P
xQ
A
B
P
E
8
9
L
Ð
Ñ
K
G
C
E
8
9
A
K
=Q
E
9
8
L
Ð
A B
B
Ñ
>
```c
{ 
    print "$\text{line}$";
}
foreach $\text{line}$ (@arr) 
    close(FILE);
$\text{arr} = \text{FILE}$;  
    open(FILE); 
    $\text{FILE} = "$\text{in}.dat$";
{
    from stdin
    
    print "$\text{line}$";
}
    $\text{arr} = \text{FILE}$; 
    foreach $\text{line}$ (@\text{arr}) 
    $\text{line} = <\text{stdin}>; 
    
    - FILE I/O
```
print "Value in dollars = 
value = pound2dollars(book)

book = 3.0;

{
    return ($EXCHANGE-RATE * pounds)
}

# refers to the first argument
$EXCHANGE-RATE = 1.54;

sub pound2dollars

- Functions:
```
print "Value in dollars: $"; 
print $dollarValue = pounds2dollars($poundValue); 

if ($0 > $ARGV[0]) {
    exit;
}

print "Specific value in pounds to convert to dollars:\n"; 

pounds on command line:

{ 
    print "$\n
[\$] $ARGV[1] \n
for ($i = 0; $i < @ARGV; $i++) {
    print "$\n
ARGV\n
@ARGV\n
number of arguments (beginning at 0) \n
- Command line arguments
```
```plaintext
{ 
    
    
    { 
    } 

    exit

    printf unknown argument (ARGV[%t])

    else

    { 
    }

    printf = ARGV$

    )

    else

    { 
    }

    printf = ARGV$

    } (d--)

    else

    { 
    }

    printf = ARGV$

    } (a--)

    else

    { 
    }

    printf = ARGV$

    } (c--)

    else

    { 
    }

    printf = ARGV$

    } (b--)

    else

    0


and balance

prints a table showing each month's principal and interest
loan - amount payment - rate
```
```c
$\text{balance} = \text{AMOUNT};
$\text{rate} = \text{RATE}/12/100;
$\text{month} = 1;

\text{print} \ "\text{Month}\%/text{Payment}\#/text{Interest}\#/text{Payment}\#/text{Balance}\n\n\";
\text{print} \ "\text{ monthly payment }: \text{PAYMENT}\n\";
\text{print} \ "\text{monthly rate } : \text{RATE}\n\";
\text{print} \ "\text{original balance } : \text{AMOUNT}\n\";

\{ 
    \text{exit} 
\}
\text{print} \ "\text{Specified - rate - amount - payment}\n\";
if ($\text{AMOUNT} == 0 || 0 == $\text{PAYMENT} || 0 == $\text{RATE} || 0 == $\text{PAYMENT})
```
```javascript
return $value;

value = (value * 100 + 0.5) / 100;

sub roundPayment

{/}

month++;

print "month";printPayment

printPayment=roundPayment;

price = roundPayment=printPayment;

price + payment + printPayment;

if (printPayment > balance) {printPayment = roundPayment=balance;

price = roundPayment=balance;

printPayment = roundPayment=balance;

while (balance > 0) {
```
<table>
<thead>
<tr>
<th>Month</th>
<th>Payment</th>
<th>Principal</th>
<th>Balance</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td>$18.31</td>
<td>$29.5</td>
<td>$90</td>
</tr>
<tr>
<td></td>
<td>$47.81</td>
<td>$29.2</td>
<td>$70.8</td>
</tr>
<tr>
<td></td>
<td>$77.01</td>
<td>$28.9</td>
<td>$51.9</td>
</tr>
<tr>
<td></td>
<td>$105.91</td>
<td>$28.6</td>
<td>$33.3</td>
</tr>
<tr>
<td></td>
<td>$134.51</td>
<td>$28.3</td>
<td>$15.0</td>
</tr>
<tr>
<td></td>
<td>$162.81</td>
<td>$28.1</td>
<td>$5.0</td>
</tr>
<tr>
<td></td>
<td>$190.82</td>
<td>$27.7</td>
<td>$33.0</td>
</tr>
<tr>
<td></td>
<td>$218.44</td>
<td>$27.2</td>
<td>$66.0</td>
</tr>
<tr>
<td></td>
<td>$245.98</td>
<td>$26.8</td>
<td>$30.0</td>
</tr>
<tr>
<td></td>
<td>$273.13</td>
<td>$26.6</td>
<td>$13.3</td>
</tr>
</tbody>
</table>

Monthly payment: $30
Interest rate: 12.5%
Original balance: $300
$ per loan pr. - r 12.5 - p 30 - a 300