Useful information for Unix

Ignacy Misztal,

November 14, 2002

General comments

Modes

One can access Unix in two modes: via telnet or via X-Windows. The first option results in a fixed-size window, and no graphics can be displayed. The X-Windows standard allows for viewing of Unix windows locally and remotely; color, graphics, resizing and cut an paste are permitted.

Important keystrokes

To exit a program, use one of the following:

`exit`	exit a session and many programs
`quit`	exit many programs
`ctrl-d`	end of file
`ctrl-c`	interrupt a program
`ctrl-z`	suspend a program

Basic commands

`pwd`	list working directory
`ls`	list files
`mkdir` *d*	make directory *d*
`cd d`	change to directory *d*
`cat` *file*	list complete *file*
`more` *file*	list *file* page-by-page
`exit`	exit the session (also exit many other programs)

file denotes one file, many files, or a wildcard

Help on all commands is via the "man" command.

`man` *command*	provide help about *command*

Different Unix systems define different backspace key. One can select a key by typing:

`stty erase` *ch*	set *ch* as erase character (could be backspace or delete)

Extensions

Files can be wildcards or can be many names.

`ls a*c`	list files starting with a and ending with c
`ls a1 a2 a3`	list files a1-a3
`ls a[1-3]`	as above
`ls a?`	list files with names starting with a and one extra character
`ls ~/reml/*`	list all files in directory *reml* in home (~) directory
`ls [!a]*`	list files that do not start with a

The same naming conventions apply to most commands.
Warning: names in Unix are case sensitive and use / slashes

Options
Most Unix commands have a great number of options. See details in the man pages. For example, the ls command can be used as

`ls -l`	list files with details
`ls -al`	list also hidden files

Informational commands

`w`	check who is active on the computer
`finger user`	find name of *user* and his (her) whereabouts (if provided)
`finger user@computer`	As above, remotely; many systems don't allow it because of security concerns
`last -10`	find the last 10 users who logged on
`df`	present statistics (including free disk space) on all filesystems
`du -k`	shows space used in current directory and its subdirectories
`whoami`	shows your userid
`hostname`	shows your computer name
`ps`	shows current processes
`ps -ef`	shows all processes
`kill -9 id`	kill process with number *id*

Copying and moving

`cp`	copy files or directories
`cp /home/user/abcd .`	copies file */home/user/abcd* to the current directory
`cp -r /home/user/abcd .`	as above but directory
`cat f1 f2 f3 >f4`	catenates files f1-f3
`mv`	as cp but moves files
`rm`	remove files or directories

`rm -r dir` remove directory *dir*

Other popular file commands

`head` print first 10 lines of a file
`tail` print last 10 lines of a files
`wc` count lines in file(s)
`grep text files` finds lines in *files* that contain *text*
`lpr` print

More exotic file commands

`sort` sort or merge
`cut` cuts specified columns from file(s)
`paste` pastes two or more files line by line
`join` joins lines of two files based on specified field
`sed` batch editor; can insert, delete and substitute lines and strings
`awk` fast programming language; can be used as one line "database" program

Editors

`vi` visual editor; works over telnet; powerful but awkward to use
`xedit` simple X-window editor
`nedit` Public domain X-Window PC-type editor
`pico` easy-to-use non-graphical editor

Linking directories and files

`ln`

link directories; or produce alternate access to a directory. To access directory `/nce/brangus/1996/data/ped` as `brped` under the current directory, type

```
ln -s nce/brangus/1996/data/ped brped
```

Running jobs in batch

One can execute jobs system in batch by using the batch command. Below, one executes program (or script) prog. Please note that input to that program needs to come from a file, `data` in this example.

```
# batch
at> prog <data
```

```
at> <ctrl-D>
warning: commands will be executed using /bin/csh
job 826136598.b at Wed Mar  6 12:23:18 1996
```

Once the program executed in the batch system completes, its output is sent by `mail`. Type `mail` to read and send mail. Type `man mail` to see `mail` options.

Progress of program in batch can be monitored by:

`ps -ef` shows list of all processes, their memory and CPU times

`top` shows graphically status of active jobs in the system.

The top command is a useful commands to see what is happening in the computer. It shows a screen like below:

```
last pid:  7477;  load averages:  0.86,  0.83,  0.84
14:33:55
33 processes:  31 sleeping, 1 running, 1 on cpu
Cpu states:  0.0% idle,  1.3% user, 57.9% kernel, 23.2% iowait, 17.5% swap
Memory: 158M real, 680K free, 184M swap, 282M free swap

  PID USERNAME PRI NICE  SIZE   RES STATE   TIME   WCPU    CPU COMMAND
 1775 ignacy   -25   19  340M  254M run    73.9H 12.20% 74.75% jaadomnlarge
 7477 ignacy    33    0 2576K 1424K cpu     0:00  0.03%  0.20% top
  104 root      33    0 1376K  320K sleep   0:39  0.02%  0.10% in.routed
  308 root      33    0 2864K  464K sleep   0:02  0.00%  0.00% rpc.ttdbserver
  213 root      33    0 2264K  552K sleep   0:01  0.00%  0.00% vold
  169 root      33    0 1456K  432K sleep   0:01  0.00%  0.00% cron
    1 root      33    0 1632K  248K sleep   0:01  0.00%  0.00% init
  175 root      -3    0 1976K  648K sleep   0:00  0.00%  0.00% nscd
  120 root      -3    0 1776K  216K sleep   0:00  0.00%  0.00% kerbd
 1770 ignacy    10    2  880K  176K sleep   0:00  0.00%  0.00% ax
 7256 root      13    0 1464K  632K sleep   0:00  0.00%  0.00% in.telnetd
 1768 ignacy    13    2  880K  176K sleep   0:00  0.00%  0.00% sh
 1769 ignacy    13    2  232K    0K sleep   0:00  0.00%  0.00% sh
  114 root      17    0 1624K  128K sleep   0:00  0.00%  0.00% keyserv
 5887 root      23    0 1816K  616K sleep   0:00  0.00%  0.00% inetd
```

It shows processes' ids, priorities, amount of memory requested and used, CPU time and name. Once top is running, type h for help or q to quit. Top can be used to reduce priority of long-running programs by typing:

```
r -20 PID
```

where PID is a process number.

Top lines of the `top` command show decomposition of the CPU time. Too much time spent in `iowait` and `swap` is an indication that programs are too many/large for the memory available.

Unix tools as filters

General syntax

```
command {options] [filename(s)]
```

if filename missing, input from standard input
output can be redirected (>) or piped (|) to other commands.

Formats of records

```
field1 field 2 field 3 ... field n
\-------field 0----------------/
```

n.p = field n, starting from character p

standard separator - space
other separators can be set by options

Commands

`cut` cuts characters or fields from records

```
cut -c -5, 10-20,25-
```
cuts characters 1-5, 10-20, 25 to end from each record

```
cut -f 1,5,7
```
cuts fields 1, 5 and 7

`paste`

```
past f1,f2,..fn
```
Pastes records f1,..,fn on the same line; space is a separator

`grep`

```
grep -n -i string file
```
lists files containing string; option -i makes matching case-insensitive, -n precedes lines by their line numbers

`sort`

```
sort f1 >f2
sort -o f1 f1
```
sort alphanumerically on all fields in ascending order; `/tmp` used for intermediate storage

```
sort +2 -3 +0 -1
```
sorting fields:
skip first 2, until 3 (resulting primary sorting filed 3)
skip first 0 until 1 (resulting secondary sorting field 1)

```
sort -T./. +.015 -0.18 +0.5 -0.7
```
sort on characters 16-18 and 6-7; intermediate directory is the current directory

other options
`-r` reverse order
`-n` numeric sort

```
sort f1 f. f3
```
sort&merge

`uniq`

retains unique lines only; file must be sorted in advance

options

`-c`	to each line add count of repeated lines
`+1 -2 ..`	take only the specified fields in determining the uniqueness (format as in sort)

`join`

joins two files (sorted) based on provided fields. See man pages for description

`tr`

```
tr ‘ ’ ‘0' f1 >f.
```
translate all spaces into zeroes

Examples

Find all lines of fortran programs that contain subroutine "adjust"

```
grep -n adjust *f
```

Prepare list of sires and number of daughters from file `f1`; sires in columns 11-18

```
cut -c11-18 f1|sort|uniq -c >f.
```

Other utilities

`sed` - batch editor: replacement, deletion, and insertion of strings

```
sed 's/1990/1992/g' f1 >f.
```
replace all 1990 into 1992

`tee`

list input and at the same time copy it to a file

```
mtdfreml|tee f1
```

`script f1`

copies all terminal traffic to file f1

```
script f1
.....
exit (or ctr-d)
```

AWK

a programming language, similar in ease but more powerful (and cryptic) than BASIC, with capabilities for line (and much more) processing

```
awk 'program' f1
awk -f program f1
```

`$0` entire line
`$1` first field
`$2` second field
`substr($0,5,10)` 10 characters of string $0 beginning with character 5

`NR` current record number
`NF` number of fields in current line

General structure of awk file

```
BEGIN     {action}
condition {action}
END       {action}
```

BEGIN & END lines not obligatory
if no condition, only {action} performed
if no {action}, list entire line

examples

awk '$1 > 100' f1
list lines in f1 where field 1 is > 100; conditions specified as in C language

```
awk '{print NR,$0}'
```
precede each record with record number

```
awk '{printf("%10s%20s%15s\n",$1,$2,$3}'
```
lists fields 1-3 in 10, 20 and 15 character space, respectively; formats are described as in C language

Problem 1

Select only first-parity records from file with the following format:

1-38 fixed fields
39 starting parity numbers
40 number of records in line
41-47 first record
48-.. later records

```
awk 'substr($0,39,1)==1{printf("%39s%1s%7s",substr($0,39,1),
                                        1,substr($0,41,7))}'
```

Problem 2

A file contains a sire in column 1-10 and a performance of its daughters in 11-15. List each sire with a consecutive number and a mean performance of its daughters. The file is sorted by sire. List the total number of sires at the end.

```
awk -f progfile
```

progfile:

```
BEGIN     {nsire=0; oldid=""; sum=0}
          {sire=substr($0,1,10);
          perf=substr($0,11,5);
                  sum=sum+perf
           if sire != oldid
               {nsire=nsire+1
               print sire,nsire,sum/nsire}
          }
END       {print sire,nsire,sum/nsire;
          print nsire}
```

Unix - Part 1

Ways to deal with long names

- file manager (in X)
- cut and paste (in X or OS/2)
- wildcards
- command editing

Wildcards

~	home directory
*	anything
?	one character
[ab]	one of a or b
[!a]	not a
[a-g]	one of a to g

Example

ls ?[a-d]*

```
/home/ignacy ls ?[a-d]*
aa          abc         backcmd*    mbox        tar.log     watorold.c  wc.c
```

Redirection and pipes

mtdfreml <input >output

Terminal will be locked

mtdfreml <input >output&

Terminal won't be locked; better submit long programs by batch

File ff:

animal_id	sire_id	WW

1,000,000 records

Find animal 1234567; don't tie the terminal

grep 1234567 ff&

or

awk '$2 == 1234567' ff&

Find progeny of 1234567 with highest WW

awk '$2 == 1234567' ff | sort +2 | head -1

```
/home/ignacy du|more
52      ./sas
2       ./.wastebasket
2       ./ignacy1
2       ./.cetables
212     ./Http/gifs
568     ./Http
548     ./ab
14      ./nsmail
2       ./back
1198    ./idled/idled-1.05
1524    ./idled
2       ./inip94/600
7212    ./inip94
2       ./.tt
30178   ./temp
110     ./wpfiles/krysia
3100    ./wpfiles
4       ./.mosaic-personal-annotations
2       ./tmp
610     ./sasuser
6       ./.spa
1114    ./meishan/dfreml/DF93/PRE
2872    ./meishan/dfreml/DF93/UNI
4036    ./meishan/dfreml/DF93/MUV
1712    ./meishan/dfreml/DF93/MUW
122     ./meishan/dfreml/DF93/MUX
400     ./meishan/dfreml/DF93/LIB
302     ./meishan/dfreml/DF93/INC
170     ./meishan/dfreml/DF93/SPA
4544    ./meishan/dfreml/DF93/bin
.....
14      ./ioffice/auxiliary
16      ./ioffice
9166    ./mtdfreml
26      ./system
2       ./.helplus
215570  .
/home/ignacy
```

Simplifying work

need to repeat the same command or string, e.g.,

/opt1/emacs/bin/emacs largedata

1. Create an alias

alias em /opt1/emacs/bin/emacs

em largedata

(works for programs only)

2. Setting the variable

set a =/opt1/emacs/bin/emacs

$a largedata

(works for names and programs)

3, Creating a symbolic link on your directory

ln -s /opt1/emacs/bin/emacs .
emacs largedata

4. Create a script

Configuration of nce.ads.uga.edu

448 Mbytes of memory
3 x 9 Gbyte applcation disks
4 x 4 Gbyte user disks
2 Gbyte for system

Names of files can be readable

To describe 1996 angus raw data:

angus.96r or angus.1996.raw

angus.1996.results.provenbulls.complete or angusrp.com

Long names too long to type. To avoid typing:
use file manager
use cut-and-paste
specify names with wildcards:

an* an*1996* a*raw [ab][nb]*1996* [!a]*

list files, and then remove the listed files

>ls very-long-name*

very-long-name1 very_long_name2
>^ls^rm

!p
!25
history

Vi editor

Full screen editor that works on every Unix machine in a non graphical environment; small subset of options shown below.
To start:

vi file for one file
vi *.f all files *.f (:n moves to next one)
vi +25 file set the cursor on line 25

Cursor mode		Type mode
	1	ESC
CTRL-F (go forward) CTRL-B (go backward) 6789 (or) hjkl		
a (insert after cursor) **A** (insert at end of line **i (**insert before cursor) **I** (insert at beginning of line	±	
x (delete 1 character) **5x** (delete 5 characters)		
dd (delete 1 line) **5dd** (delete 1 line)		
U (undelete)		
5yy (copy 5 lines)		
p (insert copied or deleted text after the cursor)		
/text (search forward) **?** (search backward) **7G** (go to line 7) **G** (go to last line)		
ZZ (save and exit)		
:q! (quit without saving)		

4. Find who logged more time last week: matt or daniel (last, wc, grep)

```
 /home/ignacy finger matt
Login name: matt                        In real life: Matt
Culbertson
Directory: /home/matt                   Shell: /bin/csh
On since May  9 11:15:30 on pts/10 from 128.192.43.68:0.0
5 hours 54 minutes Idle Time
No unread mail
No Plan.


/home/ignacy finger daniel
Login name: daniel                      In real life: Daniel
deMattos
Directory: /home/daniel                 Shell: /bin/csh
On since May  9 14:55:36 on pts/17 from nce
2 hours 13 minutes Idle Time
No unread mail
No Plan.


/home/ignacy last matt|grep May|wc
      26     260    1924
/home/ignacy last daniel|grep May|wc
      36     360    2665
```

Unix tools as filters

General syntax

command {options] [filename(s)]

if filename missing, input from standard input
output can be redirected (>) or piped (|) to other commands.

Formats of records

field1 field 2 field 3 ... field n
\-------field 0----------------/

n.p = field n, starting from character p

standard separator - space
other separators can be set by options

Commands

cut cuts characters or fields from records

cut -c -5, 10-20,25-
cuts characters 1-5, 10-20, 25 to end from each record

cut -f 1,5,7
cuts fields 1, 5 and 7

paste

paste f1,f2,..fn
Pastes records f1,..,fn on the same line; space is a separator

grep

grep -n -i string file
lists files containing string; option -i makes matching case-insensitive, -n precedes lines by their line numbers (see also egrep and fgrep)

egrep

egrep '(string1 | string2 | string3)' file
matches any of the strings

sort

sort f1 >f2
sort -o f1 f1
sort alphanumerically on all fields in ascending order; /tmp used for intermediate storage

sort +2 -3 +0 -1
sorting fields:
skip first 2, until 3 (resulting primary sorting field 3)
Skip first 0 until 1 (resulting secondary sorting field 1)

sort -T./. +.015 -0.18 +0.5 -0.7
sort on characters 16-18 and 6-7; intermediate directory is the current directory

other options
-r reverse order
-n numeric sort

sort f1 f2 f3
sort&merge

uniq

retains unique lines only; file must be sorted in advance

options

-c	to each line add count of repeated lines
+1 -2 ..	take only the specified fields in determining the uniqueness (format as in sort)

join

joins two files (sorted) based on provided fields. See man pages for description

tr

tr ‘ ’ ‘0' f1 >f.

translate all spaces into zeroes

Examples

Find all lines of fortran programs that contain subroutine “adjust”

```
grep -n adjust *f
```

Prepare list of sires and number of daughters from file f1; sires in columns 11-18

```
cut -c11-18 f1 | sort | uniq -c >f2
```

Other utilities

sed - batch editor: replacement, deletion, and insertion of strings

```
sed 's/1990/1992/g' f1 >f2
        replace all 1990 into 1992
```

tee - list input and at the same time copy it to a file

```
mtdfreml | tee f1
```

script - copies all terminal traffic to file

```
script f1
.....
exit (or ctr-d)
```

AWK

a programming language, similar in ease but more powerful (and cryptic) than BASIC, with capabilities for line (and much more) processing

awk 'program' f1
awk -f program f1

$0	entire line
$1	first field
$2	second field
substr($0,5,10)	10 characters of string $0 beginning with character 5
NR	current record number
NF	number of fields in current line

General structure of awk file

```
BEGIN     {action}
condition {action}
END       {action}
```

BEGIN & END lines not obligatory
if no condition, only {action} performed
if no {action}, list entire line

examples

awk '$1 > 100 && $2=="May"' f1
list lines in f1 where field 1 is > 100 and $2 is May;
conditions specified as in the C language

==	equal
!=	not equal
&&	and
||	or

awk '{print NR,$0}'
precede each record with record number

awk '{print log($1), $2+$3}'
write functions of fields in the record

awk '{printf("%10s%20s%15s\n",$1,$2,$3}'
lists fields 1-3 in 10, 20 and 15 character space, respectively;
formats are described as in C language

Problem 1

Select only first-parity records from file with the following format:

1-38 fixed fields
39 starting parity numbers
40 number of records in line
41-47 first record
48-.. later records

```
awk 'substr($0,39,1)==1
{printf("%39s%1s%7s",substr($0,39,39),1,substr($0,41,7))}'
```

Problem 2

data

1-10 sire no
11-15 WW

sorted by sire

1. List each sire with a consecutive number and a mean performance of its daughters.

2. List the total number of sires at the end.

awk -f progfile datafile

progfile:

```
BEGIN    {nsire=0; oldid=""; sum=0

          {sire=substr($0,1,10);
          perf=substr($0,11,5);
           sum=sum+perf
           if sire != oldid
                 {nsire=nsire+1
                 print sire,nsire,sum/nsire}
          }

END              {print nsire}
```

Homework

Select a data file with at least 10,000 but no more than 1,000,000 records. Time each command

For the homework, please use the data set /scr/ignacy1/example_file. This file contains 100,000 records, and the sire field is in columns 21-28.

Then:

1. Identify one sire; Find how many records contain that sire id (head, grep, wc)

2. Create a file that only contains sire ids (cut or awk)

3. Sort that file (sort)

4. Find number of progenies per each sire; find the sire with the largest number of progeny (uniq, sort, head)

Put the name of the file in variable $a so that it does not have to be retyped

```
/home/ignacy set a = /scr/ignacy1/example_file
/home/ignacy echo $a
/scr/ignacy1/example_file
```

Get id of the first sire

```
/home/ignacy head -1 $a|cut -c21-28
41572878
```

Command
`set b = `xxx``
puts results of command xxx in variable $b

```
/home/ignacy set b = ` head -1 $a|cut -c21-28`
/home/ignacy echo $b
41572878
```

Search the sire.

```
/home/ignacy  time grep $b $a
800784145880071590044157 2878.....
1.0u 0.0s 0:01 57% 0+0k 0+0io 0pf+0w
```
Searching took 1.0 second of users time and 0.01 wall clock time, and took 57% of processor's time.

Extract sire ids

```
/home/ignacy  time cut -c21-28 $a >a
22.0u 0.0s 0:28 78% 0+0k 0+0io 0pf+0w
/home/ignacy head -5 a
41572878
41428104
41527802
41515207
41457932
```

Sort and check

```
/home/ignacy time sort -o a a
4.0u 0.0s 0:07 52% 0+0k 0+0io 0pf+0w
/home/ignacy head -5 a
```

```
20255163
20255163
20255163
20259668
20260599
```

Summarize

```
/home/ignacy
/home/ignacy uniq -c a|sort -n -r|head -10
2551 41626813
2026 41667366
1462 41773417
1444 41650414
1253 41723741
1156 41491007
1038 41724657
1013 41810969
 881 41672151
 867 41635843
```

Sire 41626813 has 2551 progeny in the file.

Unix scripts and shells

Shells are programs that accept user's input. There are 3 popular shells in Unix: sh, csh and ksh. We dealt with csh only, and commands below apply to csh only. There are similar instructions for other shells too.

Variables

Unix variable contains text string.

```
/home/ignacy set a = /usr/local/bin

/home/ignacy echo $a
/usr/local/bin
```

Look for all files usr/local/bin/d*

```
ls $a/d*
```

To check on a variable:

```
echo $a
```

Backquotes around command(s) return text that would otherwise be displayed by the execution of the command(s).

Set $a to the first file starting in beef

```
set a = `ls beef*|head -1`
```

Commands useful for scripts

Cat
The cat command can be used to create text files.

```
/home/ignacy cat <<AA >b
>beef.5.96
>1992
>1994
AA

/home/ignacy cat b
beef.5.96
1992
1994
```

Foreach
Commands are repeated with $a assigned to consecutive elements of the list

```
foreach a (list)
.....
commands
.....
end


foreach h2 (.3 .4 .5)
blub $h2
end
```

To change all files ending with .f into .for

```
foreach a (*.f)
mv $a $aor
end
```

If statement

```
if (condition) command

if ($1 >= 235) mtdfreml

if (`ls mtdfreml.output` != ‘’ ) exit
```

operators as in awk (==, !=, >=, <=).

Script

A unix script is a text file that contains Unix instructions and that has execute permissions. In a simple case, it contains commands that otherwise would be typed on the command line

```
# This scripts runs several evaluation programs
# sequentially
#
blupedit <ed.par
bluprun <run.par
blupaccur <acc.par
blupfinish <fin.par
```

Instead of having to set up parameter files in advance, one can put them into the script

```
# Parameter file created in the script so that one does
# not have to edit parameter files separately
#
cat <<AA >ed.par
angus.sires
1970
1990
1000
AA
blupedit <ed.par
...
```

If a program needs just one line of input, it can be generated with the echo command:

```
echo 1000 .001 | bluprun
```

When one program fails, there is no need to continue

```
# This script exits before running bluprun if file
```

```
# edit.out does not exist
#
blupedit <ed.par
if (`ls edit.out` == ‘’)exit
bluprun <run.par
...
```

If script is called with parameters, e.g.,
`beefeval par1 par2 par3`
then inside the script, $1 will have value par1, $2 will have value par2 etc.

One script for many breeds

```
breedeval angus
```

Assume that:
1. Breeds are in directory /nce/breed
2. Program flows are identical

Then one can create a more universal script

```
# change to breed directory
cd /nce/$1
# create parameters in ed.par
cat <<AA >ed.par
$1.sires
1970
1990
1000
AA
# Run
blupedit <ed.par
.....
```