System Programming
(ELEC462)

Lab #2

Dukyun Nam

HPC Lab@KNU

Contents

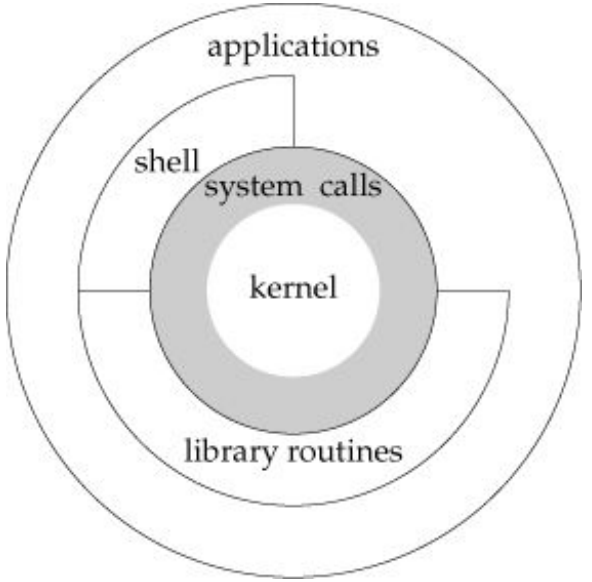
Linux/Unix Basic Commands

- Getting OS information
  - `lsb_release`: a utility to show LSB (Linux Standard Base) and Distribution information
    - `-a`: displays all of the above information.
  - `uname`: a utility to print the name, version and other details about the current machine and the operating system running on it

```
$ lsb_release -a
$ uname -a
```

Linux/Unix Basic Commands (cont.)

- What is a “shell”? (Revisit)
  - After logging in, Linux/Unix starts another program called the shell
  - The shell interprets commands the user types and manages their execution
    - The shell communicates with the internal part of the operating system called the **kernel**
    - The most popular shells are: tcsh, csh, korn, and bash
    - The differences are most times subtle
  - Shell commands are **CASE SENSITIVE**!




```
$ cat /etc/passwd | grep user_name


# Clear the history list for lab#2
$ history -c
```

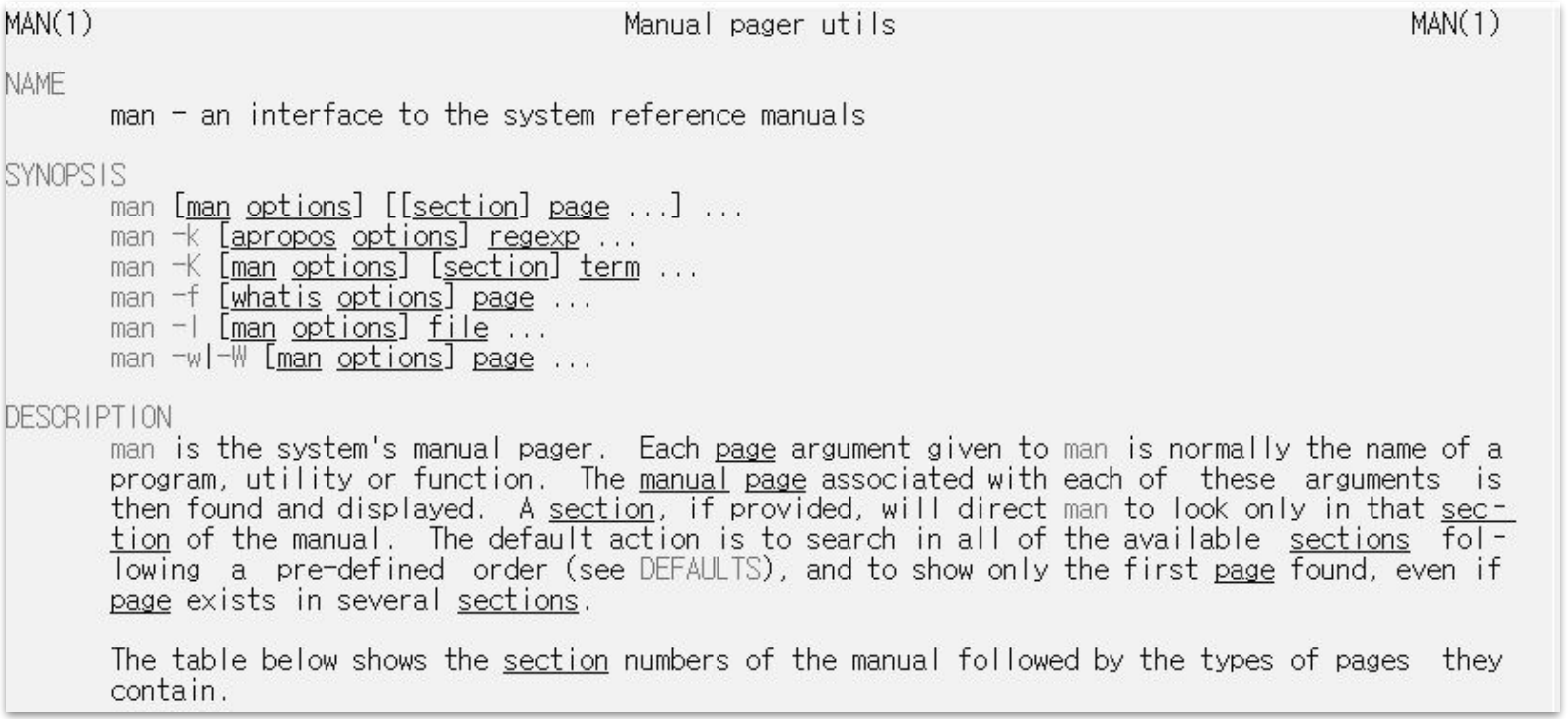
`man` Page

- Whenever you need help with a command, type "`man`" and the command name
- The manual is generally split into 8 sections.
  - Organized in the following (common to different Linux/Unix versions):

Section	Description
1	General commands
2	System calls
3	Library functions, covering in particular the C standard library
4	Special files (usually devices, those found in /dev) and drivers
5	File formats and conventions
6	Games and screensavers
7	Miscellanea
8	System administration commands and daemons

`man` Page (cont.)

```
$ man
What manual page do you want?
For example, try 'man man'.
$ man man
```



MAN(1) Manual pager utils MAN(1)

NAME
man - an interface to the system reference manuals

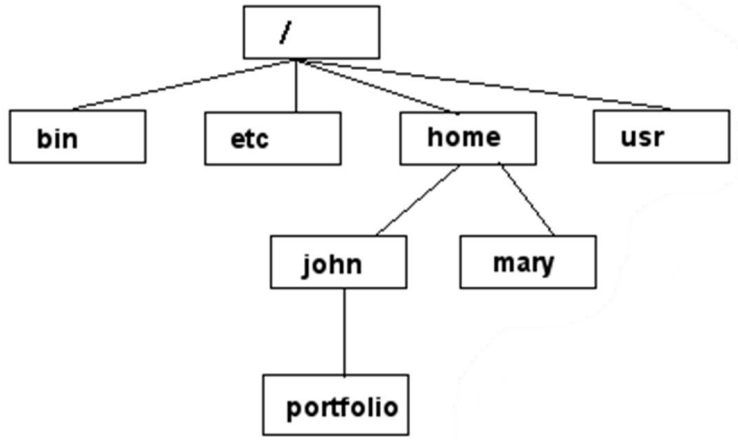
SYNOPSIS
man [man options] [[section] page ...] ...
man -k [apropos options] regexp ...
man -K [man options] [section] term ...
man -f [whatis options] page ...
man -l [man options] file ...
man -w|-W [man options] page ...

DESCRIPTION
man is the system's manual pager. Each page argument given to man is normally the name of a program, utility or function. The manual page associated with each of these arguments is then found and displayed. A section, if provided, will direct man to look only in that section of the manual. The default action is to search in all of the available sections following a pre-defined order (see DEFAULTS), and to show only the first page found, even if page exists in several sections.

The table below shows the section numbers of the manual followed by the types of pages they contain.

Listing Directory Contents

- `pwd`
  - To find your current path use “pwd”
- `cd`
  - To change to a specific directory use “cd”
  - “~” is the location of your home directory
  - “..” is the location of the directory below current one



`$ pwd`
`$ cd /etc`
`$ pwd`
`$ cd ~`
`$ pwd`
	`$ pwd`
	`$ cd /etc`
	`$ pwd`
	`$ cd`
	`$ pwd`

Listing Directory Contents (cont.)

- `ls`
  - To list the files in the current directory use “ls”
- `ls` has many options
  - -l long list (displays lots of info)
  - -t sort by modification time
  - -S sort by size
  - -h list file sizes in human readable format
  - -r reverse the order

Listing Directory Contents (cont.)

- "`man ls`" for more options
- Options can be combined: "`ls -ltr`"
  - List files by time in reverse order with long listing
- "*" can be used as a wildcard in unix/linux

```
$ ls
$ ls -al
$ ls -ltr
$ cd /usr/include
$ ls
$ ls *.h
```

Make and Remove

- `mkdir, rmdir`
  - Make and remove directories.
- `cp`
  - Copy a file into another directory or over another file.
  - Copy the entire directory into another directory.
    - Ex) 'cp –r hw1_dir /usr/share/homeworks/'
- `mv`
  - Move a file into another directory.
    - Ex) mv ./hw1.c /usr/share/homeworks/
  - Change its file name to another name
    - Ex) mv ./hw1.c hw1-renamed.c
- `rm`
  - Remove a file (or an entire directory with -r).
    - Ex) rm hw1.c (rm -r /home/*user_name*/garbage)

```
$ mkdir lab2
$ cd lab2
$ cp /usr/in[now hit a tab]
$ cp /usr/include/std[now hit 2 tabs]
stdc-predef.h  stdint.h       stdio.h
stdio_ext.h    stdlib.h

$ cp /usr/include/stdio.h testfile
$ ls -al; ls -l
$ mv testfile test
$ ls -l te[now hit a tab]
$ cp test test2
$ ls -l te[now hit 3 tabs]
$ rm test2; ls
```

Displaying a File

- `cat`
  - Dumps an entire file to standard output
  - Good for displaying short, simple files
- `less`
  - “less” displays a file, allowing forward/backward movement within it
    - return scrolls forward one line, space one page
    - y scrolls back one line, b one page
  - use “/” to search for a string
  - Press q to quit

Displaying a File (cont.)

- `head`
  - “head” displays the top part of a file
  - By default it shows the first 10 lines
  - -n option allows you to change that
    - “head -n50 file.txt” displays the first 50 lines
- `tail`
  - Same as head, but shows the last lines

```
# ~/lab2
$ cat test
$ less test
$ head test
$ head -n20 test
$ tail test
```

File Permissions

- Each file in Unix/Linux has an associated permission level
  - This allows the user to prevent others from reading/writing/executing their files or directories
  - Use "`ls -l` *`filename`*" to find the permission level of that file
- Permission levels
  - "r" means "read only" permission
  - "w" means "write" permission
  - "x" means "execute" permission
    - In case of directory, "x" grants permission to list directory contents

```
$ ls -l outline.01
-rwxr-x---     1 molay      users     1064 Jun 29 00:39 outline.01
```

```
- r w x    r w x    r w x        r: read, w: write, x:execute
  user     group    other
```

File Permissions (cont.)

- `chmod`
  - If you own the file, you can change it's permissions with "chmod"
    - Syntax: chmod [**u**ser/**g**roup/**o**thers/**a**ll]+[permission] [file(s)]

```
$ ls -l
total 4
-rw-r--r--  1 dynam dynam   76 Aug 26 18:10 hello_world.c

$ chmod a+x hello_world.c

$ ls -l
total 4
-rwxr-xr-x 1 dynam dynam 76 Aug 26 18:10 hello_world.c
```

Input/Output Redirection

- Programs can output to other programs
- Called “piping”
- `program_a | program_b`
  - program_a’s output becomes program_b’s input
- `program_a > file.txt`
  - program_a’s output is written to a file called “file.txt”
- `program_a >> file.txt`
  - program_a’s output is append to the existing file
- `program_a < input.txt`
  - program_a gets its input from a file called “input.txt”

```
$ cat test | more


$ ls -al > listing
$ cat listing
$ echo “KNU!!” >> listing
$ cat listing
$ wc -l < listing
```

Search Files with a Specific String

- `grep`
  - To search files in a directory for a specific string use “grep”

```
$ grep putc /usr/include/*.h
/usr/include/stdio.h:extern int fputc (int __c, FILE *__stream);
/usr/include/stdio.h:extern int  putc (int __c, FILE *__stream);
/usr/include/stdio.h:extern int  putchar (int __c);
/usr/include/stdio.h:extern int fputc_unlocked (int __c, FILE *__stream);
/usr/include/stdio.h:extern int  putc_unlocked (int __c, FILE *__stream);
/usr/include/stdio.h:extern int  putchar_unlocked (int __c);
/usr/include/zconf.h:#    define gzputc                z_gzputc
/usr/include/zlib.h:ZEXTERN int ZEXPORT gzputc OF((gzFile file, int c));
/usr/include/zlib.h:     Writes c, converted to an unsigned char, into the compressed
file.  gzputc

$ cat /etc/passwd | grep user_name
```

Lab #2: Submission

- Deadline: Today 11:59pm
  - 1) Exercise using a shell
    - `history > lab2_s<Your_Student_ID>.log`
      - Assume your ID is 2022000000.
      - An example command: `history > lab2_s2022000000.log`
  - 2) Write a simple C code(`sum.c`) using a `vi` Editor
    - Calculate the sum from 1 to 500
    - Output: "The sum of 1 to 500 is <*result*>."
    - `mv sum.c sum_s<Your_Student_ID>.c`
- Upload two files into LMS
  - `lab2_s2022000000.log`
  - `sum_s2022000000.c`