1 How to use Linux

Hongik University
Eunsung Jung

Disclaimer: The slides are borrowed from many sources!

UNIX history

http://www.unix.org/what_is_unix/history_timeline.html

- Originally developed in 1969 at Bell Labs by Ken Thompson and Dennis Ritchie.
- 1973, Rewritten in C. This made it portable and changed the history of OS
- 1974: Thompson, Joy, Haley and students at Berkeley develop the Berkeley Software Distribution (BSD) of UNIX
- Two main directions emerge: BSD and what was to become “System V”

Notable dates in UNIX history

- **1984** 4.2BSD released (TCP/IP)
- 1986 4.3BSD released (NFS)
- 1991 Linus Torvalds starts working on the Linux kernel
- 1993 Settlement of USL vs. BSDi; NetBSD, then FreeBSD are created
- 1994 Single UNIX Specification introduced
- 1995 4.4BSD-Lite Release 2 (last CSRG release); OpenBSD forked
- off NetBSD
- 2000 Darwin created (derived from NeXT, FreeBSD, NetBSD)
- 2003 Xen; SELinux
- 2005 Hadoop; DTrace; ZFS; Solaris Containers
- 2006 AWS (”Cloud Computing” comes full circle)
- 2007 iOS; KVM appears in Linux
- 2008 Android; Solaris open sourced as OpenSolaris

Some UNIX versions

More UNIX (some generic, some trademark, some just unix-like):

1BSD	2BSD	3BSD	4BSD	4.4BSD Lite 1
4.4BSD Lite 2	386 BSD	A/UX	Acorn RISC iX	AIX
AIX PS/2	AIX/370	AIX/6000	AIX/ESA	AIX/RT
AMiX	AOS Lite	AOS Reno	ArchBSD	ASV
Atari Unix	BOS	BRL Unix	BSD Net/1	BSD Net/2
BSD/386	BSD/OS	CB Unix	Chorus	Chorus/MiX
Coherent	CTIX	Darwin	Debian GNU/Hurd	DEC OSF/1 ACP
Digital Unix	DragonFly BSD	Dynix	Dynix/ptx	ekkoBSD
FreeBSD	GNU	GNU-Darwin	HPBSD	HP-UX
HP-UX BLS	IBM AOS	IBM IX/370	Interactive 386/ix	Interactive IS
IRIX	Linux	Lites	LSX	Mac OS X
Mac OS X Server	Mach	MERT	MicroBSD	Mini Unix
Minix	Minix-VMD	MIPS OS	MirBSD	Mk Linux
Monterey	more/BSD	mt Xinu	MVS/ESA OpenEdition	NetBSD
NeXTSTEP	NonStop-UX	Open Desktop	Open UNIX	OpenBSD
OpenServer	OPENSTEP	OS/390 OpenEdition	OS/390 Unix	OSF/1
PC/IX	Plan 9	PWB	PWB/UNIX	QNX
QNX RTOS	QNX/Neutrino	QUNIX	ReliantUnix	Rhapsody
RISC iX	RT	SCO UNIX	SCO UnixWare	SCO Xenix
SCO Xenix System V/386	Security-Enhanced Linux	Sinix	Sinix ReliantUnix	Solaris
SPIX	SunOS	Tru64 Unix	Trusted IRIX/B	Trusted Solaris
Trusted Xenix	TS	UCLA Locus	UCLA Secure Unix	Ultrix
Ultrix 32M	Ultrix-11	Unicos	Unicos/mk	Unicox-max
UNICS	UNIX 32V	UNIX Interactive	UNIX System III	UNIX System IV
UNIX System V	UNIX System V Release 2	UNIX System V Release 3	UNIX System V Release 4	UNIX System V/286
UNIX System V/386	UNIX Time-Sharing System	UnixWare	UNSW	USG
Venix	Wollogong	Xenix OS	Xinu	xMach

Linux timeline

- http://futurist.se/gldt/wp-content/uploads/12.10/gldt1210.png

UNIX Basics: Architecture

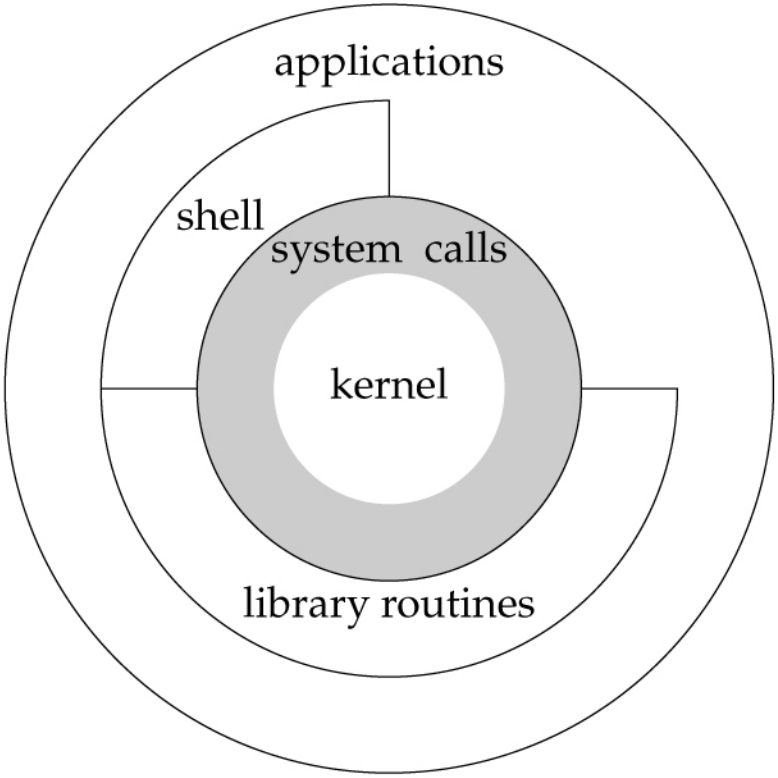


Figure 1.1 Architecture of the UNIX operating system

System Calls and Library Functions, Standards

- System Calls and Library Functions
  - System calls are entry points into kernel code where their functions are implemented. Documented in section 2 of the manual (e.g. write(2)).
  - Library calls are transfers to user code which performs the desired functions. Documented in section 3 of the manual (e.g. printf(3)).
- Standards
  - ANSI C (X3.159-1989) C89, C9X/C99 (ISO/IEC 9899), C11 (ISO/IEC 9899:2011)
  - IEEE POSIX (1003.1-2008) / SUSv4

Basic Linux Commands

Login, Password

- When connecting a UNIX system (locally or remotely)

```
login : will
password :
```

- After successful login, you will get shell prompt where you can give command input.

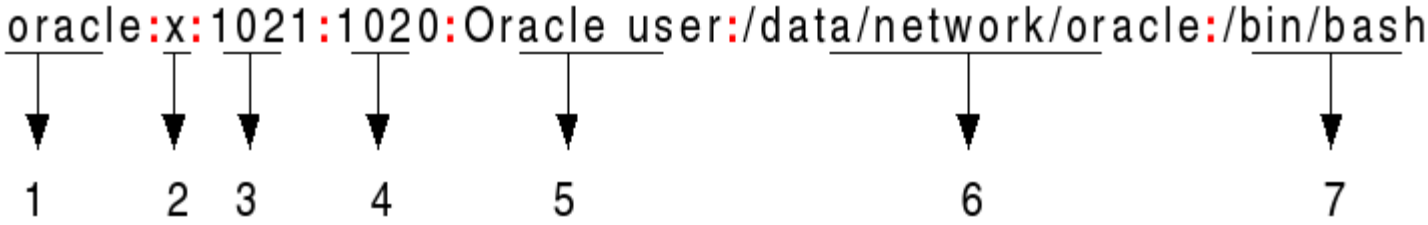
```
$
```

- To logout, type `exit`

- In GUI, it is trivial to login/logout
- Password change
  - `$ passwd`
  - Avoid dictionary words

Understanding /etc/passwd

- /etc/passwd
  - Stores essential information required during login i.e. user account information
  - It contains one entry per line for each user (or user account) of the system. All fields are separated by a colon (:) symbol. Total seven fields as follows



  - Username(1), password(2)-x means passwords are stored in /etc/shadow, user ID(3), group ID(4), user ID info(5), home directory(6), command/shell(7)
  - passwd file is readable by all users. Only root can write
    - For username-to-userid mapping

GUI

- UNIX and Linux don't incorporate the user interface into the kernel
- they let it be implemented by user level programs
- Flexible but different user interfaces exist

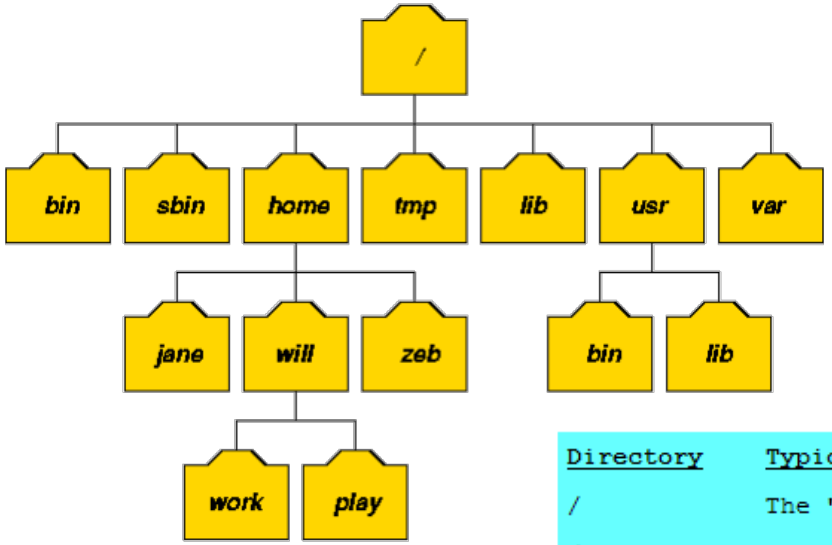
- The graphical environment primarily used with Linux is called the **X Window System**
- X also does not implement a user interface
- X only implements a window system, i.e., tools with which a graphical user interface can be implemented

- Two popular desktop managers, KDE and Gnome

UNIX File System

- Three types of UNIX files
  - Ordinary files
    - Contain text, data, program
    - Cannot contain other files or directories
    - Filename is not divided into name and extension officially
    - Up to 256 characters long
  - Directory file
    - A file that has a list of other files and directories
  - Special file : I/O device
    - Devices : for easy access to HW device, a device is dealt with as a file. e.g.) READ/WRITE for printer, network socket, …
- Links
  - A pointer to another file
  - hard link : direct pointer ex) `$ln filename linkname`
  - soft(symbolic) link : indirect pointer `ex) $ln -s filename linkname`

Directory in UNIX



< hierarchical tree structures >

Directory	Typical Contents
/	The "root" directory
/bin	Essential low-level system utilities
/usr/bin	Higher-level system utilities and application programs
/sbin	Superuser system utilities (for performing system administration tasks)
/lib	Program libraries (collections of system calls that can be included in programs by a compiler) for low-level system utilities
/usr/lib	Program libraries for higher-level user programs
/tmp	Temporary file storage space (can be used by any user)
/home or /homes	User home directories containing personal file space for each user. Each directory is named after the login of the user.
/etc	UNIX system configuration and information files
/dev	Hardware devices
/proc	A pseudo-filesystem which is used as an interface to the kernel. Includes a sub-directory for each active program (or process).

Directory

- Contains a list of files or directories and their properties/locations

- tree structure
- a parent may have many childs, and a child can have only one parent

- Path
  - absolute path : /home/bongbong/a.txt
  - relative path : usr/bin/xv

- Directory
  - Home directory, eg) cd ~bongbong
  - Current directory : . , parent directory : ..

File Access

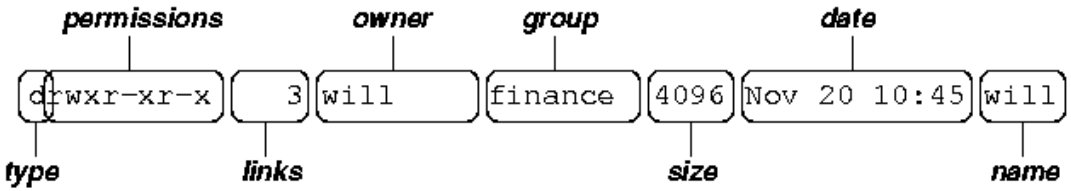
- Example : /usr/bin/xv
  1. Read root(/) directory
  2. Find the location of “usr” from “/”
  3. Read “usr” and find the location of “bin”
  4. Read “bin” and find the location of “xv”

Commands for Files and Directories

- `pwd` : prints [current] working directory
- `cd` : change directory
- `mkdir , rmdir` : create/remove a directory
- `cp, mv, rm` : copy, move, remove
- `chmod` : change permission of a file
- `cat , more` : prints text files
- `man` : manual for a command
- `ls` : list files

File properties

- `ls -l`



- ***type*** is a single character which is either 'd' (directory), '-' (ordinary file), 'l' (symbolic link), 'b' (block-oriented device) or 'c' (character-oriented device).
- ***permissions*** is a set of characters describing access rights. There are 9 permission characters, describing 3 access types given to 3 user categories. The three access types are read ('r'), write ('w') and execute ('x'), and the three users categories are the user who owns the file, users in the group that the file belongs to and other users (the general public). An 'r', 'w' or 'x' character means the corresponding permission is present; a '-' means it is absent.
- ***links*** refers to the number of filesystem links pointing to the file/directory ***owner*** is usually the user who created the file or directory.
- ***group*** denotes a collection of users who are allowed to access the file according to the group access rights specified in the permissions field.
- ***size*** is the length of a file, or the number of bytes used by the operating system to store the list of files in a directory.
- ***date*** is the date when the file or directory was last modified (written to). The -u option display the time when the file was last accessed (read).
- ***name*** is the name of the file or directory.

Link

- A pointer to another file

- Hard link to a file is indistinguishable from the file itself

  ex) `$ln filename linkname`

- Soft link (symbolic link) provides indirect pointer or shortcut to a file

  ex) `$ln –s filename linkname`

```
$ ln -s hello.txt bye.txt
$ ls -l bye.txt
  lrwxrwxrwx   1 will finance 13 bye.txt -> hello.txt
$
```

- Soft link may point to a non-existing file

Wildcard : Specifying multiple files

- UNIX shell processes this

- ‘?’ matches any one character
- ‘*’ matches any of zero or more characters
- Characters enclosed in square brackets ('[' and ']') will match any filename that has one of those characters in that position
- A list of comma separated strings enclosed in curly braces ("{" and "}") will be expanded as a Cartesian product with the surrounding characters

- he* matches any filename beginning with 'he'.
- [m-z]*[a-l] matches any filename that begins with a letter from 'm' to 'z' and ends in a letter from 'a' to 'l'.
- {/usr,}{/bin,/lib}/file expands to /usr/bin/file /usr/lib/file /bin/file and /lib/file.

File Permission

Permission	File	Directory
read	User can look at the contents of the file	User can list the files in the directory
write	User can modify the contents of the file	User can create new files and remove existing files in the directory
execute	User can use the filename as a UNIX command	User can change into the directory, but cannot list the files unless (s)he has read permission. User can read files if (s)he has read permission on them.

chmod

- Change file permission

```
 $chmod options files
ex) $chmod 600 private.txt              <- means rw-------
```

---	0
--x	1
-w-	2
-wx	3
r--	4
r-x	5
rw-	6
rwx	7

```
ex) $chmod ug=rw, o-rw, a-x *.txt     <- means rw-rw----
```

find : Finding files

```
$find directory –name targetfile –print
ex) $find . –name “*.txt” –print
```

- “ ” is necessary. Why?

- find can in fact do a lot more than just find files by name. It can find files by type (e.g. -type f for files, -type d for directories), by permissions (e.g. -perm o=r for all files and directories that can be read by others), by size (-size) etc. You can also execute commands on the files you find.

```
$find . –name “*.c” –exec wc {} ‘;’
```

- counts the number of lines in every text file in and below the current directory. The '{}' is replaced by the name of each file found and the ';' ends the -exec clause.

grep : finding text in files

- grep : general regular expression print

```
$ grep options pattern files
```

```
$ grep hello *.txt
$ grep hello `find . –name “*.txt” –print`
$ grep ^..[l-z]$ hello.txt
```

Regular Expression Syntax

- Used in grep, egrep, fgrep, vi, awk and etc

- **.** match **any** single character except <newline>
- ***** match **zero or more** instances of the single character (or meta-character) immediately preceding it
- **[abc]** match any of the characters enclosed
- **[a-d]** match any character in the enclosed range
- **[^exp]** match any character **not** in the following expression
- **^abc** the regular expression must start at the **beginning of the line** (Anchor)
- **abc$** the regular expression must end at the **end of the line** (Anchor)
- \ treat the next character literally. This is normally used to escape the meaning of special characters such as "**.**" and "*****".

- Example
  - cat the string **cat**
  - .at any occurrence of a letter, followed by **at**, such as cat, rat, mat, bat, fat, hat
  - xy*z any occurrence of an **x**, followed by zero or more **y**'s, followed by a **z**.
  - ^cat **cat** at the beginning of the line
  - cat$ **cat** at the end of the line
  - * any occurrence of an asterisk
  - [cC]at **cat** or **Cat**
  - [^a-zA-Z] any occurrence of a non-alphabetic character
  - [0-9]$ any line ending with a number
  - [A-Z][A-Z]* one or more upper case letters
  - [A-Z]* zero or more upper case letters (In other words, anything.)

Compression/Backup

- **`tar`** is used to combining files into one file (or device such as a tape) for archiving purposes

```
$ tar cvf new_file.tar dirname
$ tar cvf new_file.tar filenames
$ tar xvf new_file.tar
```

- gzip , ungzip are often used for compressing a file

```
$ gzip new_file.tar
$ gunzip new_file.tar.gz
```

Pipe/Redirection

- Output : >
- Append : >>
- Input : <
- Pipe : |

- Example

```
$ cat file1.txt file2.txt > file12.txt
$ cat file3.txt >> file12.txt
$ program < file12.txt
$ cat *.txt | grep hello
$ cat *.txt | grep hello | wc > out.txt
```

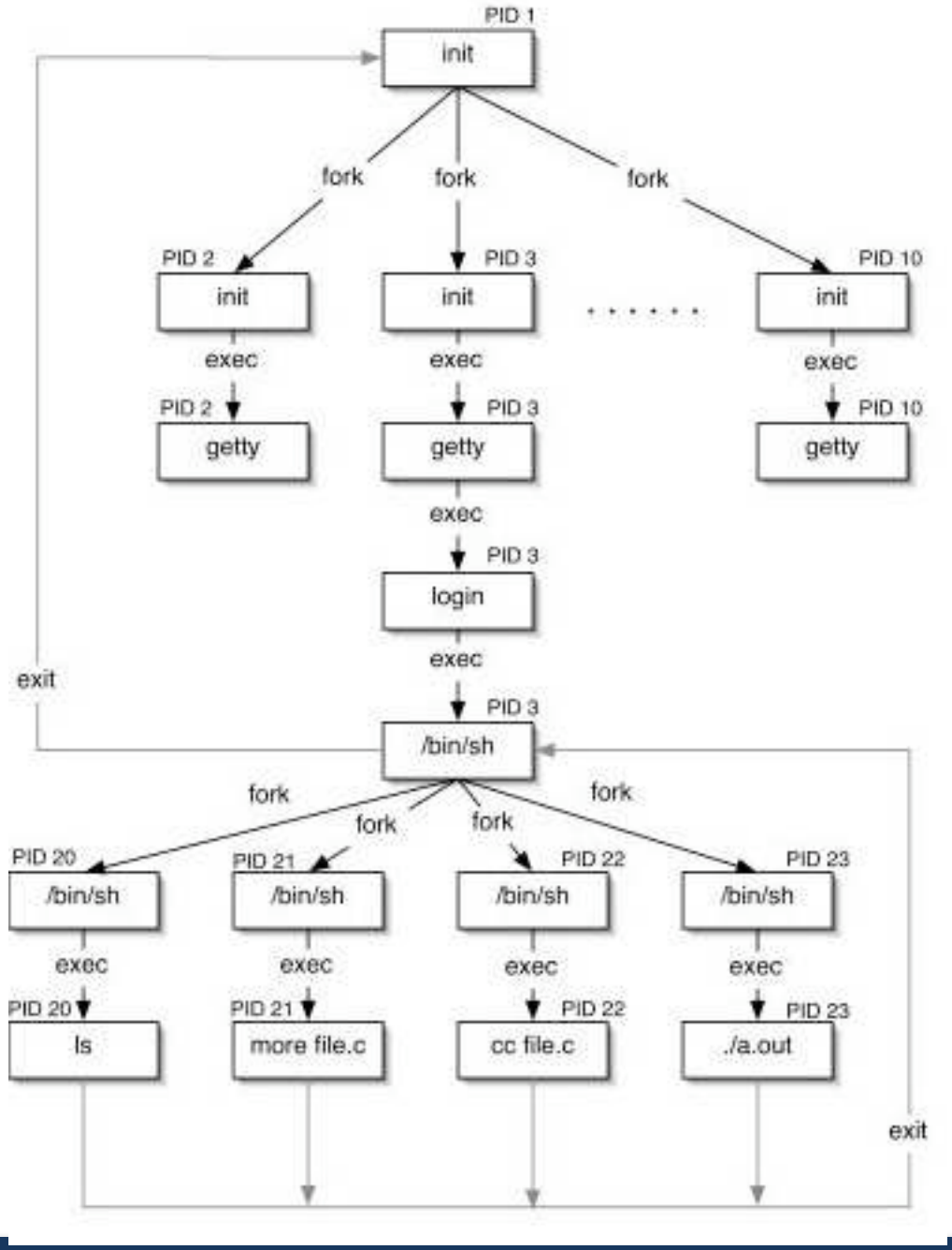
process

- Process is a program in execution
- Each time you execute a program, one or more “child” processes are created by a shell
- All UNIX process has process id or PID

process

- pstree -hapun | more

process


PID 1
init
fork
fork
fork
PID 2
init
PID 3
init
PID 10
init
exec
exec
exec
PID 2
getty
PID 3
getty
PID 10
getty
exec
PID 3
login
exec
exit
PID 3
/bin/sh
fork
fork
fork
fork
PID 20
/bin/sh
PID 21
/bin/sh
PID 22
/bin/sh
PID 23
/bin/sh
exec
exec
exec
exec
PID 20
ls
PID 21
more file.c
PID 22
cc file.c
PID 23
./a.out
exit

Background/foreground process

- UNIX shell allows multiprocessing and job control
- Jobs can be either in foreground or background
- Only one job can be in foreground at any time
- foreground job can be suspended (e.g. temporarily stopped) by pressing Ctrl-Z
- Ctrl-C : terminate foreground job
- Suspended job can continue to run by commands “fg” and “bg”
- Run a background job by appending “&” to a command (ex) find . –name “*.c” –print &

Job control

```
$ find / -print 1>output 2>errors &
    [1] 27501
$

$ jobs
[1]+  Running  find / -print 1>output 2>errors &
$

$ ps
         PID TTY          TIME CMD
       17717 pts/10   00:00:00 bash
       27501 pts/10   00:00:01 find
       27502 pts/10   00:00:00 ps

$ kill %1
   or
$ kill 27501

$ kill -9 27501      (← strong kill : -9 option sends SIGKILL signal)
```

Remote Connection

- telnet host_address
  - insecure mechnism for logging into remote machines (why insecure?)
- ssh host_address
  - Secure encrypted communication between two hosts over an insecure network.
- Ping host_address
  - Check round-trip response time between machines
  - Used for network testing, measurement and management
- ftp host_address
  - Insecure way of transfering files between machines
  - Receive (`get, mget`) , send (`put, mput`)
  - ascii (`asc`) or binary (`bin`)
  - `prompt` : interactive mode on/off
  - `cd , lcd , dir`
- `sftp, scp` : for secure file transfer