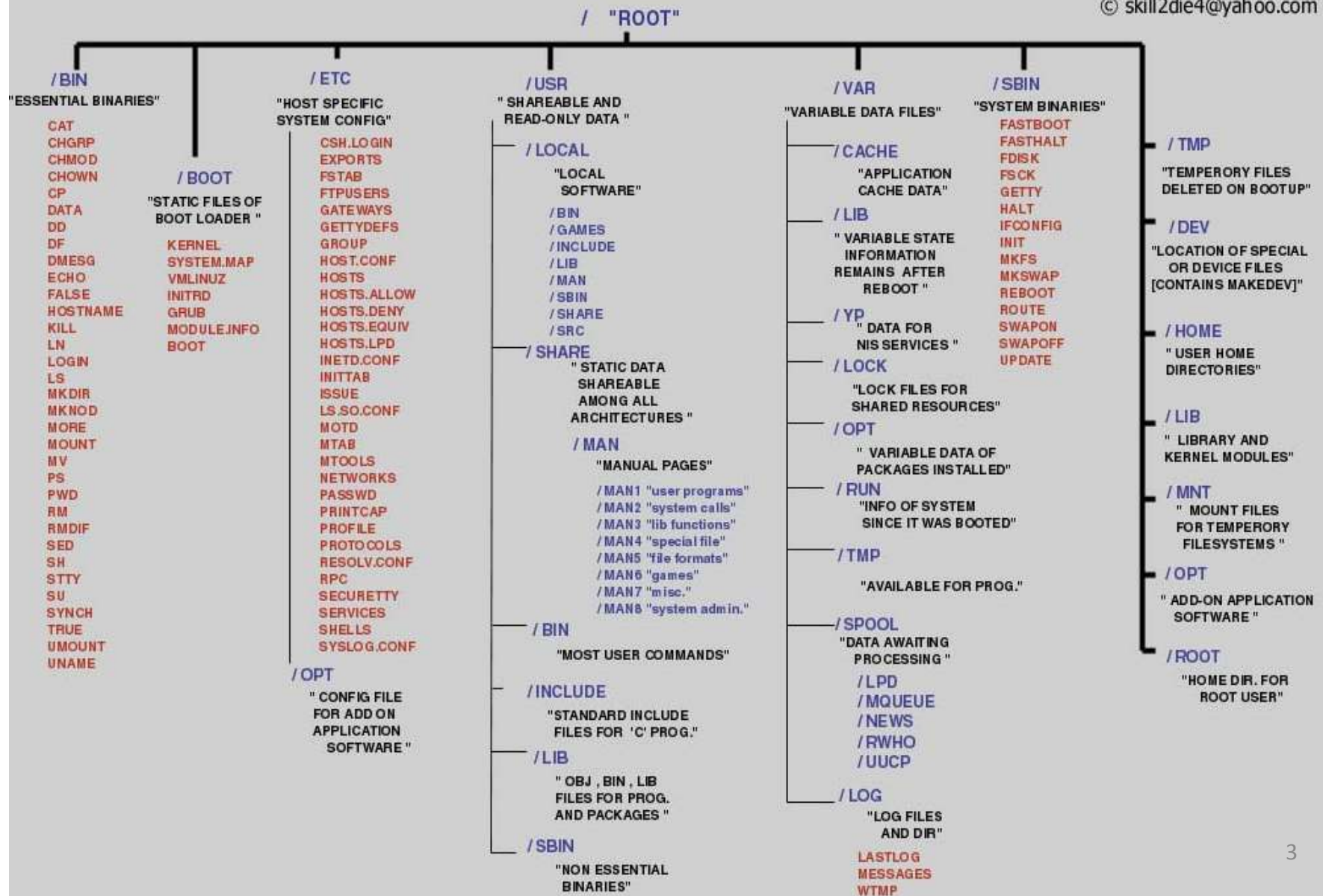


Review Linux 1

Content

- 0. Linux file system
- 1. File-directory commands
- 2. Using help with man command
- 3. Utility Commands: get information
- 4. Working with textfile
- 5. User, Group, File permissions
- 6. SHELL and Bash Shell
- 7. Install software
- 8. Managing processes

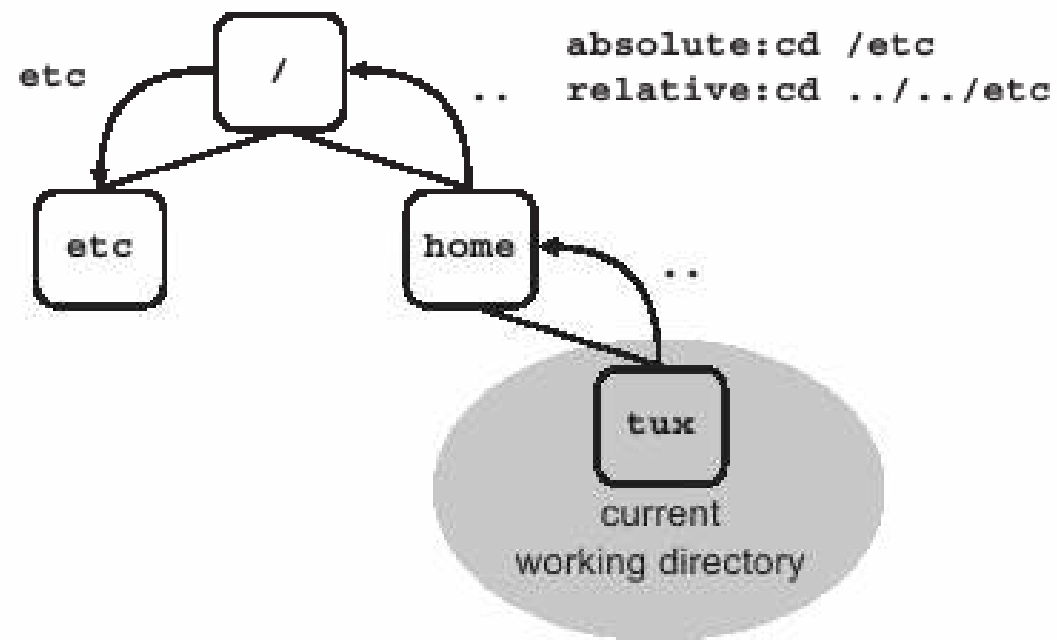
0.Linux file system



0.Linux file system



Absolute & relative paths



1.File Maintenance Commands

–ls, cp, mv, rm

```
ls -alS //hidden,long, by Size
```

```
ls *.txt
```

```
ls | more
```

```
cp file1 file2
```

```
cp file1 dir1
```

```
cp -r dir1 dir2
```

```
mv file1 file2
```

```
mv file1 dir1
```

```
mv dir1 dir2
```

```
rm file1
```

```
rm -rf dir1 : force and recursion
```

1. File Maintenance Commands

- **mkdir, cd, pwd, rmdir**

```
pwd
cd dir1
cd
cd ~
cd ..
cd /
```

```
mkdir dir1
mkdir -p dir1/dir2 //path
```

```
rmdir dir1 //only empty di
rmdir -p dir1/dir2
```

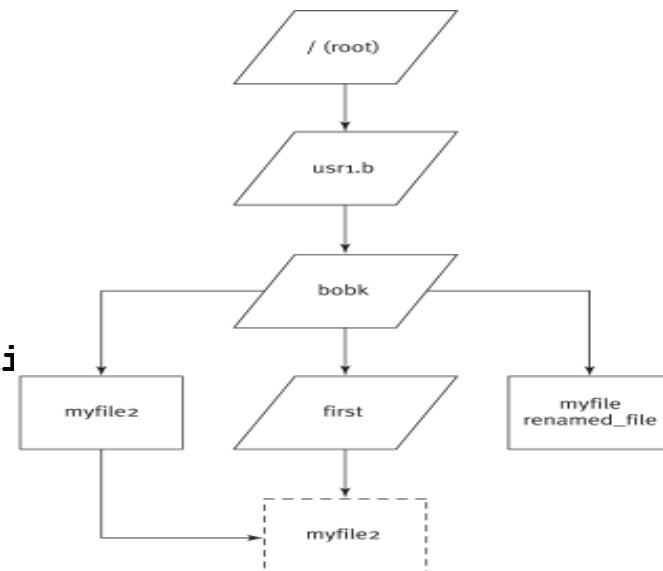
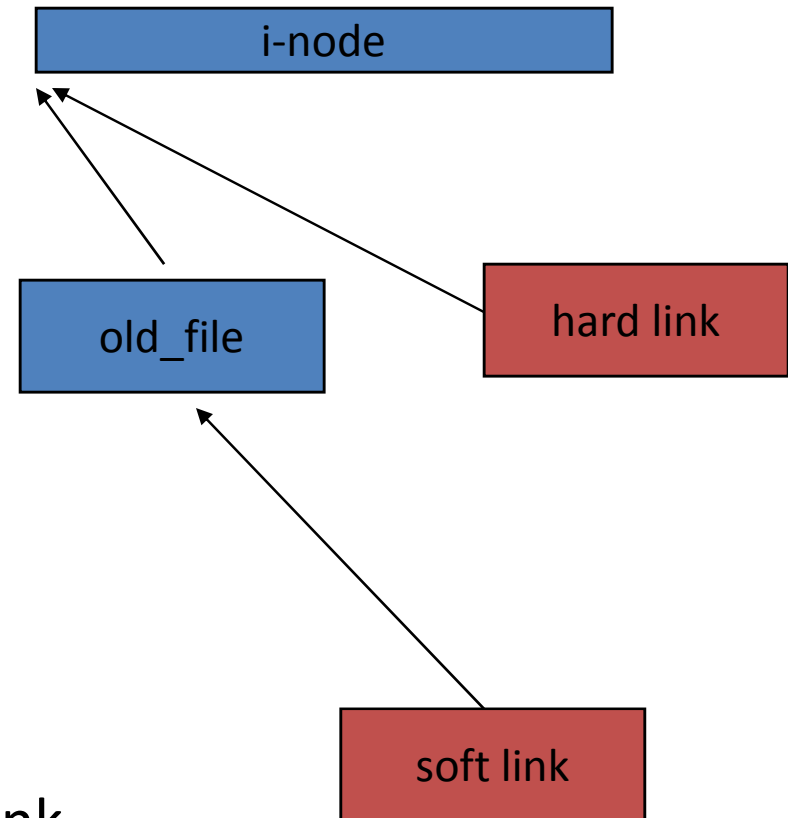


Figure 0.5 Example of a File and Directory Structure

1. File Maintenance Commands: In

- When?
- Link to file
 - hard link
 - In `/a1.txt a11`
 - soft link
 - In `-s /a1.txt a2`
 - what will happens
 - delete, change file have hardlink
 - delete, change file have softlink



2. Use Help Resources

1. Access and Use man Pages
2. Use info Pages
3. Access Release Notes and White Papers
4. Use GUI-Based Help
5. Find Help on the Web

1. Access and Use man Pages

- **man** command: online manual

Part	Contents
NAME	Name and short description of the command
SYNOPSIS	Description of the syntax
DESCRIPTION	Detailed description of the command
OPTIONS	Description of all available options
COMMANDS	Instruction that can be given to the program while it is running
FILES	Files connected in some way to the command
SEE ALSO	Hints on related commands
DIAGNOSTICS	Possible error messages of the program
EXAMPLE	Examples of calling up a command
BUGS	Known errors and problems with the command

1. Access and Use man Pages

- **man** command: with less command

Key Command	Description
<i>/expression</i>	Search forward from the current cursor position for <i>expression</i> ; matching line is displayed as first line on the screen.
<i>?expression</i>	Search backwards from current cursor position for <i>expression</i> ; matching line is displayed as first line on the screen.
n	Move to next instance of expression in the search.
N	Move to previous instance of expression in the search.
q	End display of the manual page.

1. Access and Use man Pages

- **man** command: sessions

Section	Contents
1	Executable programs and shell commands (user commands)
2	System calls
3	Functions and library routines
4	Device files
5	Configuration files and file formats
6	Games
7	Macro packages and file formats
8	System administration commands

2.Commands: search and help

- info command
 - read information of command
- apropos keyword (⇔ man -k keyword)
 - manual page header
- command --help
- whatis
 - one-line description of the command, but omits any information about options
- **find**
 - report on all classifies the named files according to the type of data files in your home directory, (ex: file *)
 - **find / -name keyword**
 - **find / -name "*keyword*"**

—

2.Commands: search and help

- **whatis**
 - To display a brief summary of a command .Shows only the **name** and **brief description** that appears near the top of a command's man page
- **whereis**
 - Locates binary, source and man page files for a command
- **which**
 - List which files are executed if the named commands are run as a command (using passwd to see the different between which and whereis)
- **apropos**
 - find in man page header
- **man -k ???**
 - find in man page header
- **locate**
 - **updatedb**
 - List which files from index database
- **find**
 - List which files directly from file system

3.Utility Commands

- Examining System Setups
 - `whoami`
 - `who`
 - `finger` (user info)
 - `cal` (display calendar)
 - `date` (display current time)
 - `uname` (display system info)
 - `clear`
 - `pwd`
 - `touch`
 - `hostname` (show system's host name)
 - How to change hostname

3. Utility Commands

- The **alias** command can be used to create pseudonyms (nicknames) for commands
- Syntax for the **alias** command is:
 - **alias [name [= string] ...]** // for Bourne, Korn, Bash shells
 - **alias l='ls -la'**
 - remove it
 - **unalias ls** // remove ls
 - **unalias -a** // remove all

TABLE 0.2 Some Useful Aliases for Various Shells
Bourne, Korn, and Bash Shells

```
alias dir='ls -la \!*' alias dir 'ls -la \!*'  
alias rename='mv \!*' alias rename 'mv \!*'  
alias spr='lpr -Pspr \!*'  
alias ls='ls -C'  
alias ll='ls -ltr' alias ll 'ls -ltr'  
alias more='pg'
```

Week1-Summary

- `cp, mv, rm, ls`
- `mkdir, cd, pwd, rmdir`
- `ln`
- `cal, date, uname, finger, hostname, clear, pwd, touch`
- `whoami, who`
- `Alias`
- `whatis, whereis, which`
- `apropos, man`
- `locate, find`

4. Working with textfile

- Viewing the Contents of Files

– **cat, more, less, head, tail**

```
cat -n file // put line # on the displayed lines
```

```
cat > myfile //create the Contents of Files:
```

```
<Ctrl-D>
```

```
more myfile
```

This is an example of how to use the cat command to add plain text to a file

4. Working with textfile

- Editing the Contents of Files

- nano

- gedit

- vi

- 3 modes (i, esc, enter) : change mode
 - (:w [filename] and :q) or (:q!)

4. Working with textfile

i or Insert	Switches vi to insert mode.
x or Delete	Deletes the character where the cursor is.
dd	Deletes the line in which the cursor is located and copies it to the buffer.
D	Deletes the rest of the current line from the cursor position.
yy	Copies the line in which the cursor is located to the buffer.
p, P	Inserts the contents of the buffer after/before current cursor position.
zz	Saves the current file and ends vi.
u	Undoes the last operation.
/ <i>pattern</i>	Searches forward from the cursor position for <i>pattern</i> .
? <i>pattern</i>	Searches backward from the cursor position for <i>pattern</i> .
n	Repeats the search in the same direction.
N	Repeats the search in the opposite direction.

:q	Ends vi (if no changes were made).
:q!	Ends vi without saving changes in the file.
:wq or :x	Saves the current file and ends vi.
:w	Saves the current file.
:w <i>file</i>	Saves the current file under the name <i>file</i> . (Note: You continue editing the original file, not the new file.)

4. Working with textfile

- Searching Content of Files: **grep**
 - Search text in a file
 - `grep apple fruitlist.txt`
 - Search text in content of files from directory
 - `grep -r root /etc`
 - Search text from result
 - `cat /etc/passwd | grep SV`
 - `rpm -qa | grep ssh`
 - `ps -A | grep gnome`

5. User- group permission

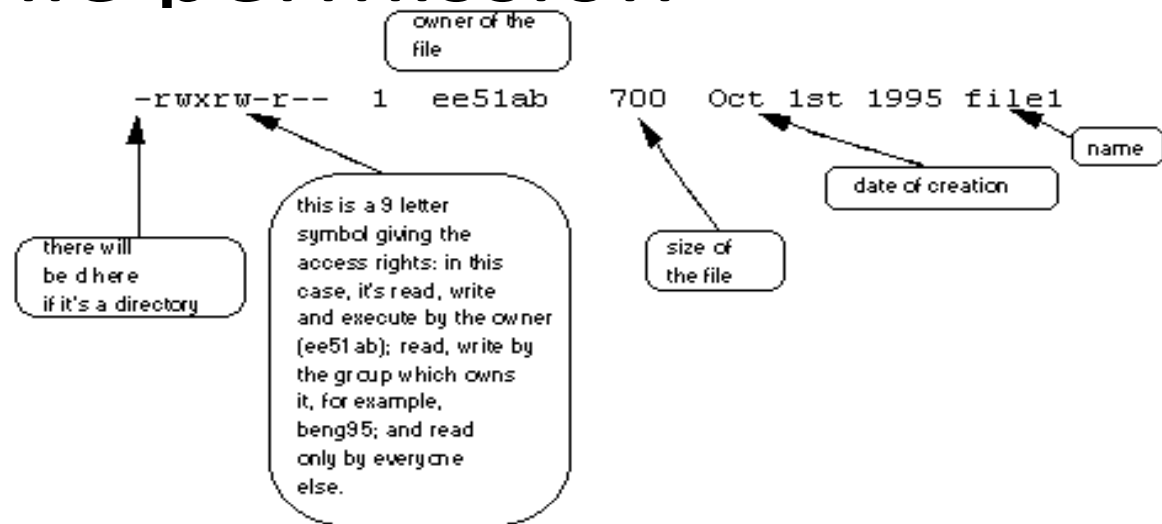
- groupadd - create a new group (groupdel)
 - groupadd g1
- useradd -- create a new user (userdel -r)
 - useradd -c Comment_username -m worker1
 - useradd -g developers worker1 (**primary group**)
 - useradd -G tester1, tester2, worker1
- usermod: change group of user
 - usermode -g G1 sv1 (change primary group)
 - usermode -G G2,G3 sv1
- passwd
- id [user]: groups of user
- groups: groups of current user
- Where?
 - [/etc/passwd](#): User names and primary groups)
 - [/etc/shadow](#): Passwords for each user
 - [/etc/group](#): Group information

5. File permissions

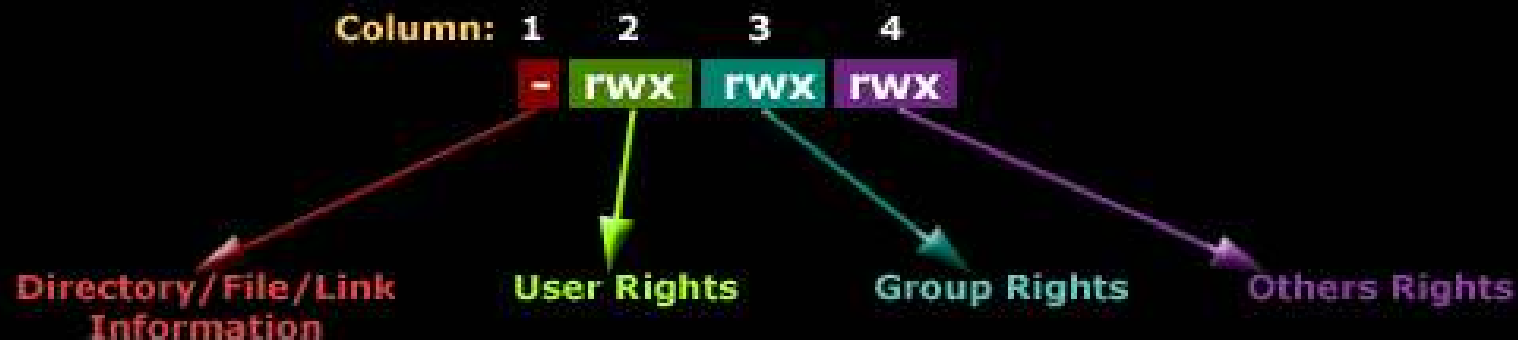
- **3 different sets:** Owner/group/others
- **3 levels of security for a file: r,w,x**
 - A hyphen :don't have that permission
- **3 levels for a directory**
 - Read(list), Write, *Enter*

```
-rwx r-xr-x joe acctg archive.sh  
- rw- rw-r-- joe acctg orgchart.gif  
- rw- rw-r-- joe acctg personnel.txt  
-d rwx r-xr-x joe acctg sales  
- rwx r-xr-x joe acctg wordmatic
```

File permission



Understanding The Linux File Permissions



While the first column defines a directory, file or link, the next 3 columns (2, 3, 4) define the permissions for the User, Group and Others (everyone else) groups.

5. File permission

- **chown - Changing Owner** of a file or directory
 - `chown user_roger file1 file2`
- **chgrp - Changing Group**
 - To change the group of a file or directory
 - `chgrp group_mt file1 file2`
- **chmod**
 - (see later)
- **using `-R` for recursion in directory**
- **Notes**
 - Only the superuser may change the owner of a file.
 - The owner of a file may change the group of the file to any group of which that owner is a member.

5. File permissions

- **chmod command**
 - *chmod who=permissions filename*
 - who: u (user),g(group),o (other),a(all)
 - example: `chmod og=rw publicity.html`
 - *chmod who+(-) permissions filename*
 - *using for add or remove permission*
 - example: `chmod go-w publicity.html`: remove write permission for the **g**roup and **o**thers
 - *Numeric Permission*
 - View: `ls -l`
 - *using binary number: rwx (421)*
 - example: `chmod 644 file.htm`: read/write by the owner and only read by everyone else (`-rw-r--r--`)

5. File permissions

- Example
 - create /public/pub.txt
 - create 3 groups: g1, g2, g3
 - create 3 users: u1(g1, u2, u3) ,u2(g2), u3(g3)
 - Set permissions for /public: a=rwx
 - pub.txt (u2, g2) 740
 - using accounts: u1, u2, u3 to see file content

5. File permissions

- **3 more bits**
 - **Sticky bit**
 - **SUID**
 - **SGID**

5. File permissions

- **Sticky bit**
 - Mainly set on directories
 - `chmod +t /directory`
 - only people who can **rename** or **remove** any file in that directory
 - the file's owner
 - the directory's owner
 - the superuser

5. File permissions

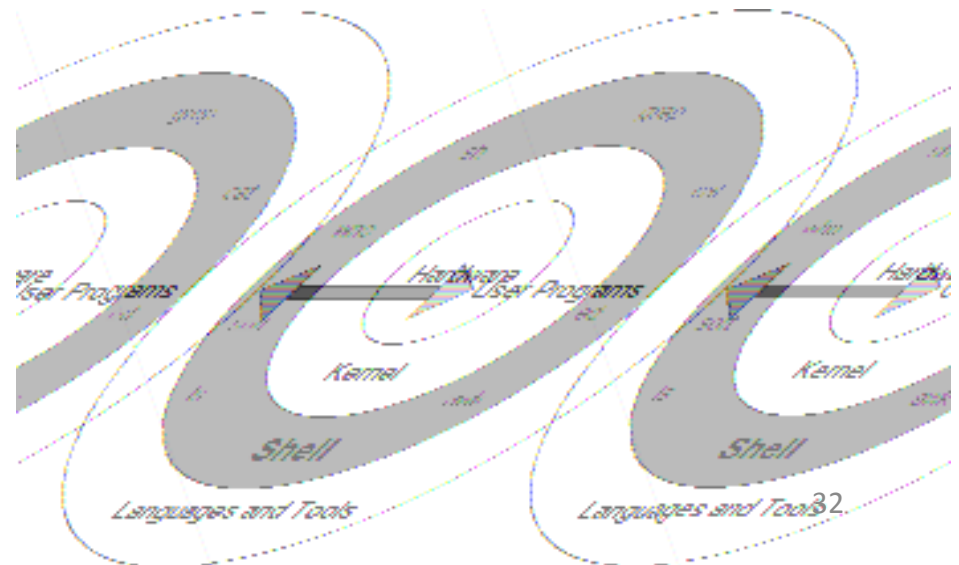
- SUID
 - execute tasks with other user permission
 - `chmod u+s file`
 - example : change password
 - Note: security
- SGID
 - On file: execute tasks with other group permission
 - `chmod g+s freddy`
 - example: chat app (wall, write, mesg,mail)
 - On directory
 - new files and subdirectories created within it to inherit its groupID, rather than the primary groupID of the user who created the file

Week 2: Summary

- nano, gedit, vi (cat >)
- useradd,passwd,groupadd
- usermod
 - 3 level: ugo
 - 3 permission: rwx
 - 3 ways to use (=,+-,(ugoa) binary number)
- chown, chgrp, chmod
- Sticky bit, SUID,SGID
- write, wall, mail

6. UNIX Shell

- The **UNIX/LINUX** shell is a program
 - important part of a Unix system.
 - interface between the user & UNIX kernel
 - starts running when you log
 - interprets the commands that you type.
 - Before runs commands, determines whether cmd is
 - **internal**
 - **own code** is executed
 - **external**
 - **search** for a file that has the **name** of the cmd by **searching** several **directories**
 - the **search path** is determine in the **shell variables** **PATH** (or **path** in the TC shell)
 - You can **view** the **search path** by using **echo \$PATH**



6. What a UNIX/LINUX shell

- There are many different shells
- All shells derive many of their features from the Bourne shell, or `/bin/sh`
- Every Unix system needs the Bourne shell to function correctly
- `bash` is the default shell for most Linux distributions
- When a user is created, many startup files are placed in their home directory: `.bash_logout`, `.bash_profile`, `.bashrc` ...

Other Shell

Shell	Location on System	Program (Command) Name
Bourne	/bin/sh	sh
Bourne Again	/bin/bash	bash
C shell	/bin/csh	csh
TC shell	/bin/tcsh	tcsh
Korn shell	/bin/ksh	ksh
Z shell	/bin/zsh	zsh

`chsh -` change the default shell

`chsh -l` available shells with a path

`echo $SHELL`

From here , bash shell only..... =>

Shell Startup File Elements

- What goes into a shell startup file?
 - The Path
 - What should be in the path?
 - The Prompt
 - What does a reasonable prompt look like?
 - Aliases?
 - What makes a good alias?
- These are the types of questions you need to think about when thinking about putting anything in your startup files.

PATH and path variables

- HOME variable
- PATH variable for search path
 - echo \$PATH
 - set PATH: *is the same for 2 line below???*
 - PATH=/sbin:\$PATH
 - PATH=\$PATH:/sbin
- Change the PATH for each user?
 - *~*/.bash_profile (first log) , *~*/.bashrc (new shell)
 - Change it to the following:
 - PATH=\$PATH:\$HOME/bin:/sbin
- How:change the PATH for every shell?
 - /etc/bashrc, /etc/profile
 - Example: exit wrong PATH\$ in /etc/bashrc

PS1 variabile

- `PS1='type something! '`
- Some of the codes that can go in the prompt are:
 - `\d` the date (day-of-the-week month day)
 - `\h` the hostname (without the domain)
 - `\n` start a new line
 - `\u` the current username
 - `\w` the working directory
 - `PS1='[\u] [\n][\w]'`

6. In what order are they executed

- `/etc/profile`
 - sets up user environment information for every user, is executed when you first log in
- `/etc/bashrc`
 - executed for every user who runs the bash shell, time a bash shell is opened, can be overwritten in each user's `~/.bashrc` file
- `~/.bash_profile`
- `~/.bashrc`
 - best location to add environment variables and aliases
- `~/.bash_logout`
 - clear screen

6. In what order are they executed

- In order to change the main default files ,
/etc/profile and /etc/bashrc: you must be root.
- Users control the information in the
~/.bash_profile, ~/.bashrc, and ~/.bash_logout
files.
- If a system administrator sees a good change
in a user's account, they might want to
consider changing the defaults for all users.

6. Shell script

- Scripts are collections of commands that are stored in a file. (backup, get information...)
- Using vi, gedit, nano
- Running script
 - bash scriptname
 - scriptname (need permission)

```
# My first script
echo "Hello TTCNTT!"
echo Kernel version
uname -r
```

8. Compress file

- gzip (gunzip), bzip2 (bunzip2) ,zip (unzip) : compress and replace
 - gzip (bzip2) mydata.doc
 - zip mydata.zip mydata.doc
- tar: archiving, support gzip (-z) & bzip2 (-j)
 - archiving: tar -cvf myarchive.tar mydirectory/
 - de-archiving: tar -xvf myarchive.tar mydirectory/
 - combine with gzip
 - compress: tar -zcvf pics.tar.gz *.jpg
 - decompress: tar -zxvf pics.tar.gz

9. Install software

- From Add-Remove
 - Have internet
- **From .rpm package**
- Using apt-get or yum
 - Have internet or Linux DVD Setup
- From source code
- yum, apt-get
- **From .deb package**
 - (Debian) convert using 3th tools: alien

9. From .rpm package

- RPM Package Manager (RPM) is a powerful command line driven package management system
 - installing
 - uninstalling
 - verifying
 - querying
 - updating
- Each package consists of: an archive of files, information about the package: version, a description...
- Where to download free package
 - <http://freshmeat.net>, <http://rpm.pbone.net>
 - http://www.icewalkers.com/rpm/cat/amusements_games/fedora-core-9.html
 - rpmseek.com

9. From .rpm package

- When you use RPM for installing the software package, RPM checks
 - if your system is suitable for the software the RPM package contains
 - figures out where to install the files the package provides
 - installs them on your system
 - adds that piece of software into its database of installed RPM packages

9. From .rpm package

- **Installing and upgrading,remove**
 - `rpm -i(U) software-2.3.4.rpm`
 - `rpm -e software`
 - other option: v , h (`rpm -ivh soft.rpm`)
- **Error: failed dependencies**
 - RPM complains about failed dependencies even when the needed program does exist
 - you've used some other method for installing a certain program
 - `rpm -ivh software-2.3.4.rpm --nodeps`
- **Querying the RPM database**
 - `rpm -q software`
 - `rpm -qa //list all package`
 - `rpm -qa | grep kde`
- Example: `gnome-games-2.22.1.1-4.fc9.i386.rpm`
- Example: install VMware-Tools

9. From .rpm package

- `rpm -ivh {rpm-file}`
 - Install the package
- `rpm -Uvh {rpm-file}`
 - upgrade
- `rpm -ev {package}` , `rpm -ev --nodeps {package}`
- `rpm -qa`
- `rpm -qf {/path/to/file}`
 - Find out what package a file belongs to i.e. find what package owns the file

9. From source code

- Compile
 - Need GCC – C++
- Read **INSTALL/README** first
 - ./configure
 - make
 - make install
- Example
 - Install john

week3-Summary

- Bash Shell
 - Variable: PS1, SHELL, PROMPT
 - chsh, chsh -l
 - ~/.bashrc, ~/.bash_profile
 - /etc/profile, /etc/bashrc
- Simple copy file from host XP -> VMware
 - Bridged
 - ifconfig, ping, smb://
 - Nat
- Install software using RPM
 - rpm -ivh (-e), --nodeps
 - rpm -q, (-qa)
- Install software from source code
 - ./configure, make, make install
 - README, INSTALL

Remote login – exchange information

- start sshd service
- using putty to connect remote
- using WinScp to copy file
- exchange information
 - write
 - wall
 - mesg
 - mail

7.Copy file from host to VM

- In-Direct: From Host
 - Through network
 - simple with sharing folder: samba: smb://
 - Using WinSCP
- Direct
 - from USB pen disk: depend on OS
 - from CDROM, DVD: mount CDROM
- Through vmware
 - Add harddisk from VM workstation (mount ntfs /dev/sda3)
 - Shared folder from VM workstation (/mnt/hgfs)
 - Using iso file mount cdrom disk through vmware device
 - drag-drop file from host<->VM using vmware-tools
 - install vmware-tools
 - Check kernel-level

10. Managing Process

- List all Process
- Kill process
- Background – Foreground process
- nice - renice
- Command
 - ps, pstree, kill, killall, top, nice, renice, fg, bg
- Schedule process
 - at daemo
 - cron daemon

10. Managing Process

- Running program
- Form in a Tree
 - Top is init
 - PID with PPID

10. Managing Process

- ps command
 - Show only your process
 - -ef
 - -el
 - -efH
 - -u + username
- Be familiar with ps output
 - UserName, PID, PPID, TTY

State

R for runnable

S for sleeping

T for stopped

Z for a zombie

orphan process

Flag

State

Schedule

F	S	UID	PID	PPID	C	PRI	NI	ADDR	SZ	WCHAN	TTY	TIME	CMD
4	S	501	24116	24110	0	80	0	-	1217	wait	pts/1	00:00:00	bash
0	R	501	24163	24116	2	80	0	-	1153	-	pts/1	00:00:00	ps

10. Managing Process

- Example: orphan process

- cat > orphan
sleep 600
\$./orphan &
[1] 18901
\$ ps -f

UID	PID	PPID	C	STIME	TTY	TIME	CMD
666	16625	16622	0	15:58	pts/3	00:00:00	-ksh
666	18901	16625	3	17:51	pts/3	00:00:00	/bin/sh ./orphan
666	18905	18901	1	17:51	pts/3	00:00:00	sleep 600
666	18906	16625	0	17:51	pts/3	00:00:00	ps -f

- \$ kill 18901

- \$ ps -f

UID	PID	PPID	C	STIME	TTY	TIME	CMD
666	16625	16622	0	15:58	pts/3	00:00:00	-ksh
666	18905	1	0	17:51	pts/3	00:00:00	sleep 600
666	18913	16625	0	17:51	pts/3	00:00:00	ps -f

- [1] + Terminated ./orphan

- Example: Zombie process

- Make script with loop

10. Managing Process

- top: show app using most RAM
- kill (+ PID)
- killall (+name)
- nice
 - run program
 - default =10
 - -20 ..19 (nice priority)
 - ps -l: show nice priority
- renice
 - an existing program
 - user only increase, root can decrease

10. Managing Process

- top: show app using most RAM, CPU

PID	USER	PR	NI	VIRT	RES	SHR	S	%CPU	%MEM	TIME+	COMMAND
2390	ttcntt	20	0	38440	13m	8712	R	1.3	5.3	3:25.71	gnome-system-mo
1887	root	20	0	49340	13m	5264	S	0.7	5.6	2:51.43	Xorg
21174	root	20	0	2120	1076	210	R	0.7	0.4	0:02.27	top

10. Managing Process

- kill command
 - send signal to process
 - many different signals that can be sent
 - generally most interested are [SIGTERM](#) and [SIGKILL](#)
 - However, many programs do not implement for SIGTERM (default signal) (example: stop find command)
 - Send SIGTERM
 - kill 1234, kill -TERM 1234, kill -15 1234 : cleanup
 - Send SIGKILL
 - kill -KILL 1234, kill -9 1234: tells a process to stop execution immediately without cleanup
 - SIGTERM (15) signal should be attempted first
 - other signal:
 - SIGINT (ctrl+C): interrupt
 - SIGTSTP (ctrl+Z): stop
 - SIGQUIT (ctr+\): quit
- pkill, killall: base on process name, use with signal

10. Managing Process

- Background process
 - if process take a long time...
 - using &
 - find / -name txt &
 - CTRL-z, and using
 - fg (continues a stopped job by running it in the foreground) +jobid
 - bg (place a job in background) (+job id)
- jobs
 - lists the jobs that you are running in the background and in the foreground
- example: make 5 second looping program

10. Managing Process

- nice
 - run a command with modified priority from -20 (highest priority) to 19 (lowest priority)
 - nice: print default priority
 - **\$nice -n 5 ls**
 - Increment the priority value of the ls command by 5 and run
 - **#nice -n -5 ls**
 - Decrement the priority value of the ls command by -5 and run
- renice
 - alters the priority
 - **\$renice +1 123**
 - **\$renice -1 123**
- Note: only root can decrease nice value

10. Managing Process

- at daemon (atd)
 - schedules a command to be ran at a particular time
 - at now + 2 minutes, at 6:32 (see page 584)
 - press command + Ctrl+D
 - Without display on terminal

10. Managing Process

- cron daemon
 - repeat. Check crontab every minute
- System cron daemon (root)
 - /etc/crontab
 - /etc/cron.hourly (daily, weekly, monthly)
 - /etc/cron.d
- User cron daemon
 - using crontab –e for edit (* * * * * command)
 - in /var/spool/cron (with the same user name)