Operating Systems

Alireza Sharifi

Overview

- The core components of a modern operating system
- How to use the command line interface (CLI)
- Basic operations to navigate the file system
- File permissions for users and groups
- User account management
- Software management

- An operating system and its use

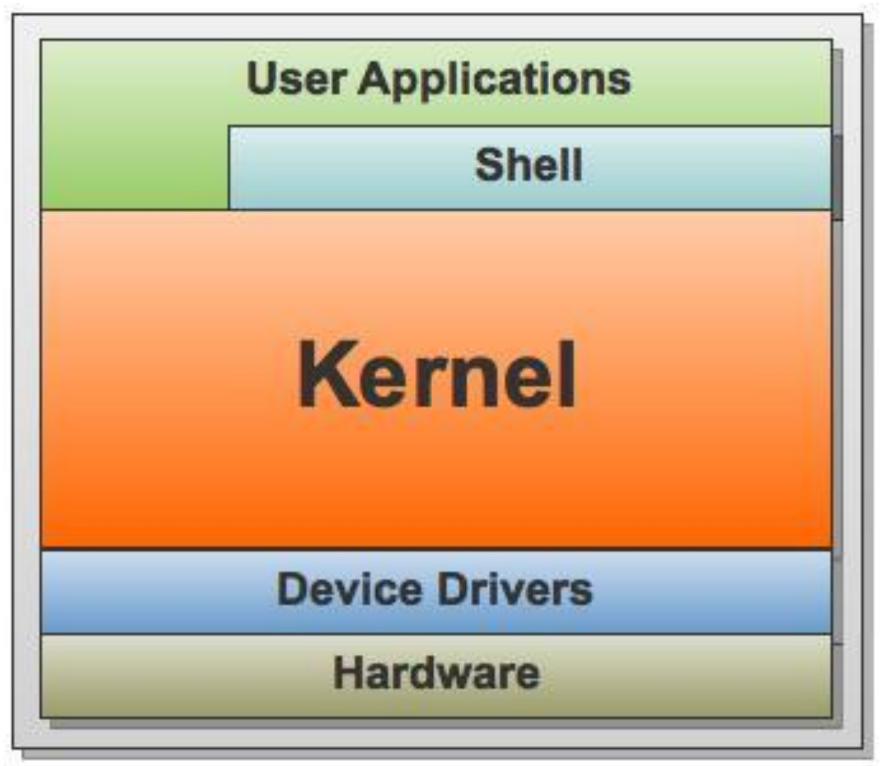
Operating system

- Definition
  - Software that manages computer hardware and provides common services to user applications
    - Application developers can ignore the details of the underlying hardware when developing applications
      - Greatly simplifies application development

Operating system

- Components
  - Kernel
    - Controls hardware devices
    - Manages memory
    - Executes code on the computer's CPU
    - Hides details of underlying physical hardware from applications
  - Shell
    - Text-based program that allows the user to interact directly with the kernel

Operating system structure



Shell

- Primary interface for system administrators
  - Direct access to OS structures
    - In plain English
  - Low network bandwidth needs
  - Scripting (automation) capabilities
- Windows
  - PowerShell runs as a separate program
    - Relatively new introduction to the OS (Windows 7, Server 2008)
- Unix
  - Ready on OS start-up
  - Easily accessible from Windows
  - Focus of course

Bash prompt information

- [alice@sunshine   usr]$

Current privileges
$ => ordinary privileges
# => root privileges

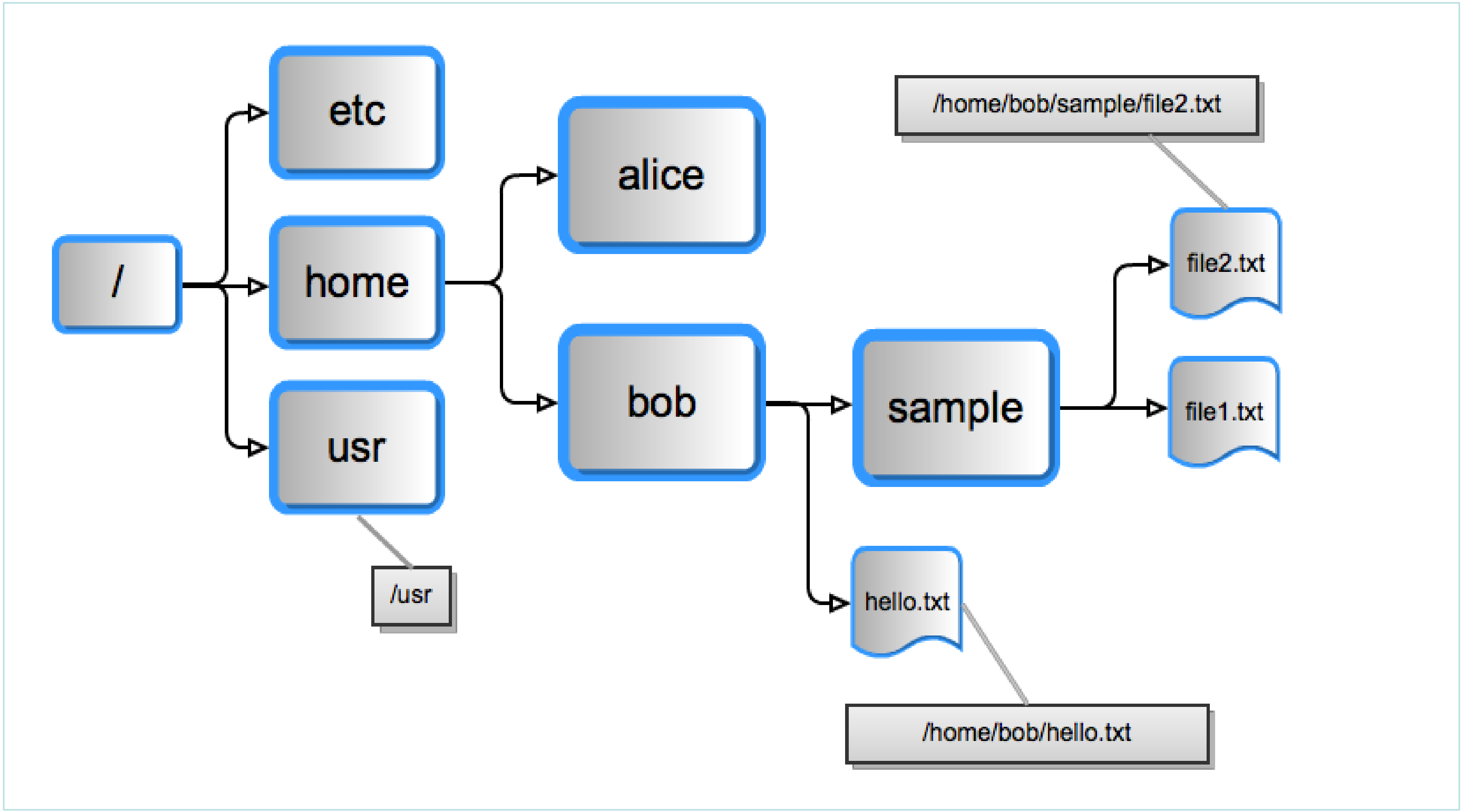
Current folder

Computer connected to (in a data center, you may be connected to one of thousands of computers)

Logged in user name

Common operations

- File navigation
- File management
- File content viewing and editing
- Search
- Access control
- User management
- Access control lists
- File permissions
- Software installation and updates


/
etc
home
usr
/usr
alice
bob
sample
hello.txt
/home/bob/hello.txt
file2.txt
file1.txt
/home/bob/sample/file2.txt

File system navigation

- Filesystem root
  - Top of the file hierarchy
  - Represented as a single slash
    - /
- Path
  - Location of a file or directory in the hierarchy
  - Representation
    - Two ways
      - Absolute
      - Relative

Type	Description	Examples
Absolute	• Exact location of file or folder being referenced • Includes each directory above the final one, up to the file system root	/usr/tmp/hello.txt /home/bob/sample/file2.txt
Relative	• Location of the file or folder in relation to the current directory	hello.txt sample/file2.txt

Case sensitivity

- UNIX file systems are case sensitive
  - /usr ≠
    - /Usr, or
    - /USR, or
    - /usR
    - etc
  - /home/bob/sample/file2.txt ≠
    - /home/Bob/sample/file2.txt, or
    - /home/BOB/sample/file2.txt, or
    - /Home/bob/sample/file2.txt
    - etc

Moving around

- pwd
  - Present Working Directory
  - e.g.
    - [alice@sunshine ~]$ pwd
    - [alice@sunshine ~]$ /home/alice
- cd
  - Change Directory
  - e.g.
    - [alice@sunshine ~]$ cd /usr/bin (absolute path specified)
    - [alice@sunshine bin]$ pwd
    - [alice@sunshine bin]$ /usr/bin
- ~
  - Indicates home directory

Commands, options, arguments

- On the previous slide
  - cd – command
  - /usr/bin – argument
- Command
  - Direction to the computer to perform something
- Argument
  - Relevant information to the computer to help execute the command, e.g.
    - If you want to cd, it helps for the computer to know which folder you wish to go to
  - Most commands have meaningful defaults, e.g.
    - cd without arguments will take you to your home directory

Moving up

- e.g.
  - [alice@sunshine Desktop]$ pwd
  - [alice@sunshine Desktop]$ /home/alice/Desktop
  - [alice@sunshine Desktop]$ cd ..
  - [alice@sunshine ~]$ pwd
  - [alice@sunshine Desktop]$ /home/alice
- ..
  - Represents folder just above current folder
- .
  - Represents current folder
    - Used shortly

Listing folder contents

- ls
  - [alice@sunshine ~]$ ls
  - Desktop Documents Downloads hello.txt Music Pictures Public Templates Videos

  - To distinguish folders from files
    - [alice@sunshine ~]$ ls -F
    - Desktop/ Documents/ Downloads/ hello.txt Music/ Pictures/ Public/ Templates/ Videos/
    - Folders marked by trailing /

Commands, options, arguments

- Options
  - Indication to the command to behave in a certain way
  - Modify default behavior of the command
  - e.g.
    - ls vs ls –F
      - Show folders in a certain way
- Options are also called flags, or switches
  - Start with a –
    - No space between – and flag
  - Usually one letter (e.g. –F)
    - But can be full words (e.g. -debug)

Commands, options, arguments

- Options can be combined, e.g.
  - ls –f –l
  - For simplicity, single letter options may be written together, e.g.
    - ls –fl
- Most commands have many, many options, e.g.
  - ls
    - -aAbcCdeEfFghHilLmnopqrRstuvVx1
- Some option combinations are very popular, e.g.
  - ls –al (shown on next slide)
    - -a: also show hidden files
    - -l: long listing (show details)

Commands, options, arguments

```
[alice@sunshine share]$ cd /home/shared/
[alice@sunshine shared]$ ls -al
total 28
drwxr-xr-x. 6 root root               4096 Jan 28 19:10 .
drwxr-xr-x. 8 root root               4096 Jan 28 19:06 ..
drwxr-xr-x. 2 root accounting_grp  4096 Jan 28 19:07 accounting
drwxr-xr-x. 2 root engineering_grp 4096 Jan 28 19:06 engineering
drwxr-xr-x. 2 root marketing_grp   4096 Jan 28 19:07 marketing
-rw-r--r--.    1 root root                   22 Jan 28 19:10 README
drwxr-xr-x. 2 root sales_grp          4096 Jan 28 19:06 sales
```

Command autocomplete

- Tab key
  - Will complete commands and arguments to the extent possible

- Try typing
  - cd
  - ls –al p<TAB>

- If multiple options
  - Auto-complete to the extent possible
  - Double <TAB> displays available options

Shell expansions

- GUI is convenient
  - But CLI has its own tricks
- Shell expansions (wildcards) simplify command entry
- 3 wildcards

?	Matches characters	re?d matches reed and read but not reads
*	Matches any zero or more characters	re* matches red, reed, read and reads
[x..y]	Matches a range of letters or numbers	re[a,e]d matches reed and read but not red

Shell expansion examples

- [alice@sunshine Expansion]$ ls

```
goodbye.doc  heap.txt  helicopter.txt  hello.doc  hello.txt
help.txt
```

- [alice@sunshine Expansion]$ ls *.doc

```
goodbye.doc  hello.doc
```

  - Only .doc files shown

- [alice@sunshine Expansion]$ ls he?p.txt

```
heap.txt  help.txt
```

File management

- mkdir
  - Creates directories

```
[alice@sunshine work]$ mkdir new_directory
[alice@sunshine work]$ ls -aF
./  ../  new_directory/
```

- rmdir
  - Removes directories

```
[alice@sunshine work]$ rmdir new_directory/
[alice@sunshine work]$ ls -aF
./  ../
```

Copying and moving files

- General syntax
  - <cmd> <source> <target>
- Copy
  - cp
    - [alice@sunshine work]$ cp hello.txt hello_world.txt
    - [alice@sunshine work]$ ls -aF
    - ./ ../ hello.txt hello_world.txt
- Move
  - mv
    - [alice@sunshine work]$ mv hello_world.txt HELLOWORLD.TXT
    - [alice@sunshine work]$ ls -aF
    - ./ ../ hello.txt HELLOWORLD.TXT

Recursion

- If folder contains folders and files and folders within those folders
  - cp only operates at the top level files
    - mv is always recursive
  - Recursion causes copies to be made within folders
  - Useful to copy directories
  - e.g,
    - [alice@sunshine alice]$ ls -F
    - Desktop/ Documents/ Music/ Pictures/ Public/ Videos/
    - [alice@sunshine alice]$ cp -r Desktop/ Desktop-copy
    - [alice@sunshine alice]$ ls -F
    - Desktop/ Desktop-copy/ Documents/ Music/ Pictures/ Public/ Videos/

Removing (deleting) files and folders

- Remove command
  - rm

```
[alice@sunshine ~]$ cd ~/Desktop-moved/
[alice@sunshine Desktop-moved]$ ls -aF
./  ../  notes.txt  readme  sample_file1.mp3
[alice@sunshine Desktop-moved]$ rm notes.txt
[alice@sunshine Desktop-moved]$ ls -aF
./  ../  readme  sample_file1.mp3
```

-i option

- cp, mv and rm are invasive commands
  - No recovery possible
- -i option
  - Adds interactivity
  - Warning appears if the operation will delete an existing file
- e.g.

```
[alice@sunshine Desktop-moved]$ rm -i readme
rm: remove regular file `readme'? n
[alice@sunshine Desktop-moved]$ cp -i sample_file1.mp3
readme
cp: overwrite `readme'? n
```

Removing folders

- rmdir works with empty folders
- rm –r <target> to delete folders with content

```
[alice@sunshine alice]$ ls -F
Desktop/  Desktop-moved/  Documents/  Music/  Pictures/  Public/
Videos/
[alice@sunshine alice]$ rm -r Desktop-moved/
[alice@sunshine alice]$ ls -F
Desktop/  Documents/  Music/  Pictures/  Public/  Videos/
```

- Warning: Probably most lethal command in your arsenal
- rm –r /
  - ?

Viewing files

- Most system administration files are text files
- less <filename>, e.g.

```
[alice@sunshine shared]$ less /usr/share/doc/openssl-1.0.0/FAQ
```

  - View file contents one screen at a time
  - Includes powerful search features
    - /word
      - Search towards the end of the file for the first occurrence of the word
    - ?word
      - Search towards the beginning of the file for the word
  - Search can be repeated
    - n
      - search for the next occurrence of the word
    - N
      - Search for the previous occurrence of the word

Viewing portions of files

- Sometimes quick view of files is useful
  - e.g. top of file to see if it is the file you need
    - head
  - Or, bottom of file to see new log entries
    - Useful when editing software configuration
    - tail
- [alice@sunshine shared]$ head /etc/passwd

```
root:x:0:0:root:/root:/bin/bash
bin:x:1:1:bin:/bin:/sbin/nologin
daemon:x:2:2:daemon:/sbin:/sbin/nologin
… [7 more lines]
```

Viewing portions of files

- [alice@sunshine shared]$ tail /etc/group

```
sales_grp:x:504:
engineering_grp:x:505:
… [8 more lines]
```

- Default output has 10 lines
  - -n option specifies number of lines
- [alice@sunshine shared]$ tail -n5 /etc/group

```
sales_grp:x:504:
engineering_grp:x:505:
marketing_grp:x:506:
eric:x:507:
accounting_grp:x:508:
```

Searching for files

- find
- e.g.

```
[alice@sunshine ~]$ find / -name httpd.conf
/etc/httpd/conf/httpd.conf
```

- Takes two arguments
  - First argument
    - Directory to search in
  - Second argument
    - File to look for
    - Wildcards can be used (?, *)

Find command

- Slightly unusual compared to other commands
  - Also extremely powerful and versatile

- Second argument is called an expression
  - Many operators defined
- E.g.
  - Find empty files owned by alice

```
[alice@sunshine ~]$ find /home -user alice -empty
/home/alice/.bashrc
```

Access control

- Files need protection
  - Confidentiality
  - Also, safety
    - End users should not be able to delete server configuration files by accident
- Two mechanisms available
  - File permissions
    - Very traditional
    - Universally agreed and standardized
  - Access control lists (ACLs)
    - Fine grained
    - Relatively new and limited tool support

Viewing file permissions (ls –al output)

```
drwxr-xr-x. 2 root marketing_grp   4096 Jan 28 19:07 marketing
-rw-r--r--.    1 root root                   22 Jan 28 19:10 README
```

Col 1	**2**	**3**	**4**	**5**	**6**	**7**

Column	Description	Example
1	File/ directory permissions	drwxr-xr-x
2	Number of file system “hard” links	2
3	File/ directory user ownership	root
4	File/ directory group ownership	marketing_grp
5	File/ directory size (in bytes)	4096
6	Modification time stamp	Jan 28 19:07
7	File/ directory name	marketing

File permissions in column 1 (ls –al)

- First column in ls –al output has 10 characters, e.g.
  - drwxr-xr-x

- Actually two sets of information

  - First character
    - File type

  - Remaining 9 characters
    - File permissions

Symbol	**Type**
d	Directory
-	Regular file
b	Block/ special file
c	Character/ special file
l	Symbolic link
p	Named pipe
s	Socket

Permissions

- 9 characters
  - 3 sets of 3 characters each
- e.g.

r w x	r - x	r - -
Owner	Group	World

- File owner
  - First 3 characters
- File owner group
  - Second set of 3 characters
- Everybody else (world)
  - Last set of 3 characters

Permissions (contd.)

- r
  - Indicates permission to read file (using less, head, tail etc)
- w
  - Indicates permission to edit file (using vi or other editors)
- x
  - Indicates permission to execute file (commands)
- e.g.
  - r w x r – x r - -
- Owner can read, write and execute file
- Group can read and execute file (but not edit)
- Everybody else can only read the file

Permissions (Contd.)

- Execute permission for a user is specified by the third character
  - x indicates execute permission is available
  - - indicates user/ group/ world cannot execute the file

- Other values are possible for this position
  - s (setuid/ setgid)
    - File runs with permission of owner/ group, not user executing the file
    - Often used by developers to simplify testing
      - Security hazard
  - T (sticky bit)
    - Users may write, but cannot move or delete files in this directory

Octal notation

- Many administrators prefer to use a shorthand to represent file permissions
  - Supported by most commands

r	w	x
4	2	1

- Permissions interpreted as a 3-bit binary number
  - Read permission = 4
  - Write permission = 2
  - Execute permission = 1

- Permissions add up
  - 5 = 4 + 1 = Read and execute permission
    - Equivalent to r-x

Octal notation examples

- 755
  - Owner
    - Read, write, execute (4 + 2 + 1)
  - Group
    - Read, execute (4 + 1)
  - World
    - Read, execute (4 + 1)
- 644
- 664
- 660
- 777

Changing permissions

- What if you want to add or remove permissions
  - E.g. Group does not have write permissions in Engineering directory
  - Say we want to allow group members write permissions on the directory

- chmod
  - The chmod command can update permissions on files and folders
  - Generally requires super user (root) privileges
    - Owner can also change permissions

Gaining super-user privileges

- su
  - The su command confers super-user privileges

```
[alice@sunshine shared]$ su -
Password: EnterTheRootPassword
[root@sunshine ~]#
```

- Note the change in prompt
  - $ → #
- # indicates super-user privileges
  - System assumes you know what you are doing
    - Minimal interactivity and confirmations
- Be very, very careful at # prompt

Before chmod

```
[root@sunshine ~]# cd /home/shared
[root@sunshine shared]# ls -laF
total 28
drwxr-xr-x. 6 root root              4096 Jan 28 19:10 ./
drwxr-xr-x. 8 root root              4096 Jan 28 19:06 ../
drwxr-xr-x. 2 root accounting_grp    4096 Jan 28 19:07 accounting/
drwxr-xr-x. 2 root engineering_grp   4096 Jan 28 19:06 engineering/
drwxr-xr-x. 2 root marketing_grp     4096 Jan 28 19:07 marketing/
-rw-r--r--.  1 root root               22 Jan 28 19:10 README
drwxr-xr-x. 2 root sales_grp         4096 Jan 28 19:06 sales/
```

After chmod

```
[root@sunshine shared]# chmod 775 engineering
[root@sunshine shared]# ls -laF
total 28
drwxr-xr-x. 6 root root             4096 Jan 28 19:10 ./
drwxr-xr-x. 8 root root             4096 Jan 28 19:06 ../
drwxr-xr-x. 2 root accounting_grp   4096 Jan 28 19:07 accounting/
drwxrwxr-x. 2 root engineering_grp  4096 Jan 28 19:06 engineering/
drwxr-xr-x. 2 root marketing_grp    4096 Jan 28 19:07 marketing/
-rw-r--r--. 1 root root               22 Jan 28 19:10 README
drwxr-xr-x. 2 root sales_grp        4096 Jan 28 19:06 sales/
```

Access control lists

- Allow fine-grained application of permissions
  - getfacl
  - setfacl

```
[root@sunshine shared]# getfacl README
# file: README
# owner: root
# group: root
user::rw-
user:alice:rw-
user:bob:rw-
group::---
group:devs:r--
mask::rw-
other::---
```

File ownership

- chown
  - Change ownership
- chgrp
  - Change group

- Example on next slide

chown and chgrp example

```
[root@sunshine shared]# cd /home/shared
[root@sunshine shared]# chown dave README
[root@sunshine shared]# chgrp sales_grp README
[root@sunshine shared]# ls -laF
total 28
[…]
drwxrwxr-x. 2 root engineering_grp 4096 Jan 28 19:06 engineering/
drwxr-xr-x.  2 root marketing_grp   4096 Jan 28 19:07 marketing/
-rw-r--r--.  1 dave sales_grp         22 Jan 28 19:10 README
drwxr-xr-x.  2 root sales_grp       4096 Jan 28 19:06 sales/
```

Software installation and updates

- Linux/ Unix software is often called a package
  - Refers to all files of the application bundled together as one file
  - Comparable to .iso files used in Windows software

- Software is managed by applications called package managers, e.g.
  - apt (Debian), rpm (Redhat), pkgutil (Solaris) etc
  - CentOS uses yum
    - YellowDog Updater, Modified
- Open source software is available online
  - Repositories

Homework

- Research Project Now Available