Shell Scripting

- This lesson aims at introduce you to the shell programming language

- Topics are organized in four slides sets and cover concepts related to:

  - 1) Unix account and filesystem
  - 2) The command line interpreter (shell)
  - 3) Commands for text handling
  - 4) Shell scripts

Why shell scripting?

- Flexibility to interact with the Operating System services
- Automate repetitive file management operations
- Write programs instalation scripts
- Start up programs and gluing their input/output

Why shell scripting?

- Used from/integrated to Debugging, Container Management, Build Automation tools and IDEs ej. adb, Docker, Maven, Android Studio
- Inspect and analyze operating system/application logs to diagnose failures
- Windows Subsystem for Linux (WSL) allows Microsoft Windows users to use Linux Bash shell
- And the list could continue...

First things, first

- Use a Unix-based system. Meaning for Windows users, DOWNLOAD A LINUX DISTRO. Recommended:
  - Ubuntu (Debian-based) http://www.ubuntu.com
  - Linux Mint (Ubuntu-based) http://www.linuxmint.com
  - CentOS (RedHat-based) http://www.centos.org/
  - OpenSUSE http://es.opensuse.org/
  - elementaryOS (Ubuntu-based) http://elementaryos.org/
- If You want to install it in your machine:
  - Create a bootable USB using:
    - Lili http://www.linuxliveusb.com/
    - Unetbootin http://unetbootin.sourceforge.net/
  - Burn a CD or DVD.
- You can also use a virtualization app like Virtual box. For Virtual Machines, a lightweight operating system is recommended: E.g. Lubuntu or Xubuntu

Unix Accounts and the Filesystem

Sistemas Operativos I

Unix Accounts

- To access a Unix system you need to have an *account*

- A Unix account include:
  - username and password
  - userid and groupid
  - home directory
  - shell

Logging In

- To log in to a Unix machine you can either:
  - sit at the *console* (the computer itself)
  - access via the net (using telnet, rsh, ssh, kermit, or some other remote access client).
- The system prompts you for your username and password.
- Usernames and passwords are case sensitive!

Home Directory

- A home directory is a place in the file system where the account files are stored (similar to C:\Users\username in Windows)
- A *directory* is like a Windows folder (more on this later)
- Many unix commands and applications make use of the account home directory (as a place to look for customization files)

Shell

- When you log in to a Unix system the program you initially interact with is your shell

- A Shell is a unix program that provides an interactive session - a text-based user interface also known as CLI (command line interface)

- There are a number of popular shells that are available. Find out your configured shell by typing “echo $SHELL” in a console or terminal window

Some Simple Commands

Here are some simple commands to get you started:

- ***ls*** lists file names (like DOS dir command)
- ***who*** lists users currently logged in
- ***date*** shows the current time and date
- ***pwd*** print working directory
- ***df*** disk usage information
- ***ifconfig*** network interfaces
- ***dmesg*** messages from devices
- ***lspci*** connected devices

Files and File Names

- A *file* is a basic unit of storage (usually storage on a disk)
- Every file has a name
- Unix file names can contain any characters (although some make it difficult to access the file)
- Unix file names can be long! (up to 255 unicode characters depending on the filesystem)

File Contents

- Each file can hold some raw data
- Unix does not impose any structure on files
  - files can hold any sequence of bytes
- Many programs *interpret* the contents of a file as having some special structure
  - text file, sequence of integers, database records, etc.

Directories

- A *directory* is a special kind of file - Unix uses a directory to hold information about other files

- We often think of a directory as a container that holds other files (or directories).

More about File Names

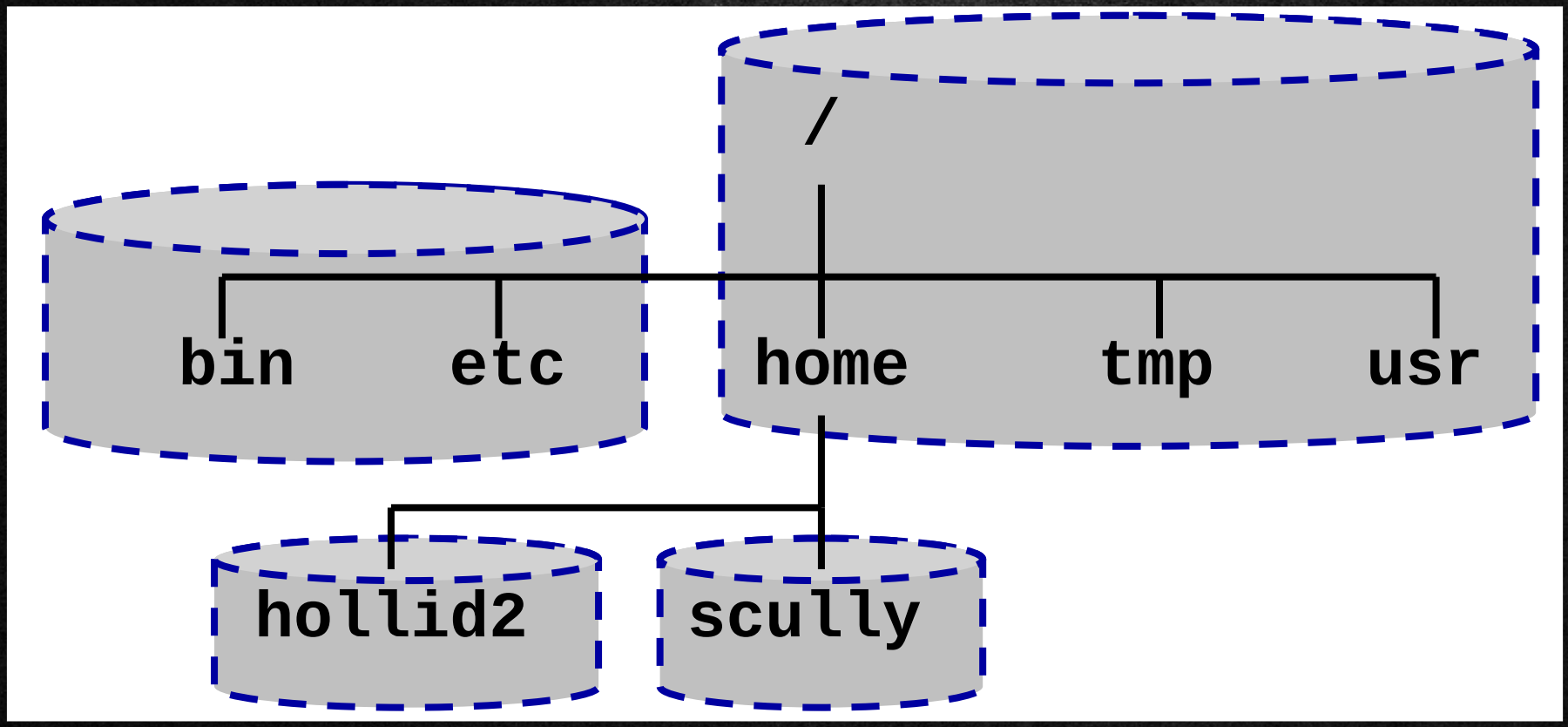
- Review: every file has a name
- Each file *in* the same directory must have a unique name
- Files that are in different directories can have the same name.

Unix Filesystem

- The ***filesystem*** is a hierarchical system of organizing files and directories.
- The top level in the hierarchy is called the "root" and *holds* all files and directories.
- The name of the root directory is "**/**"

Disk vs. Filesystem

- The entire hierarchy can actually include many disk drives.
  - some directories can be on other computers
- **df**: shows what disk holds a directory.



Pathnames

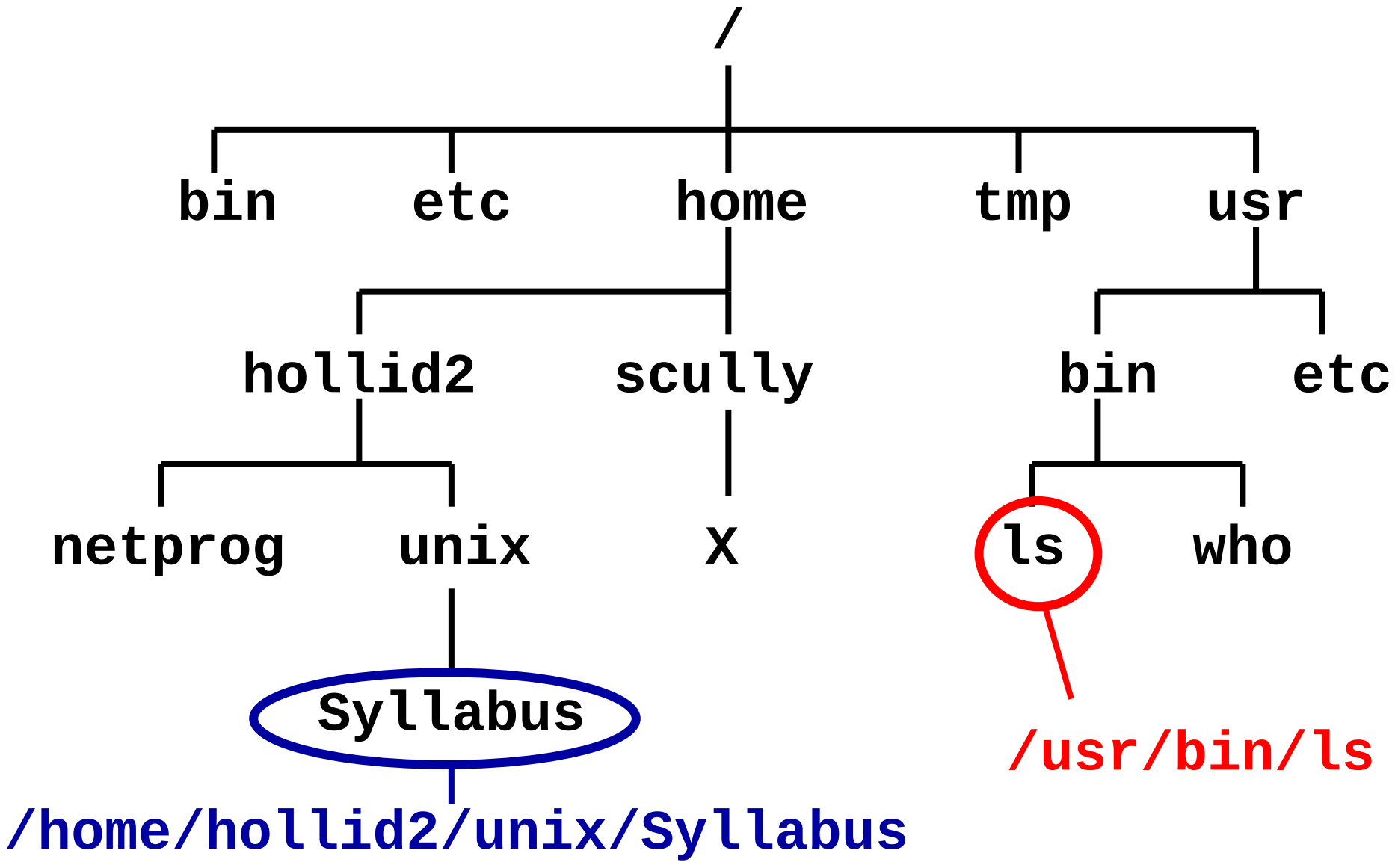
- The *pathname* of a file includes the file name and the list of directory names up to the root
- The pathname of every file in a Unix *filesystem* is unique

Pathnames (cont.)

- To create a pathname you start at the root (so you start with "/"), then follow the path down the hierarchy (including each directory name) and you end with the filename

- In between every directory name you put a "/"

Pathname Examples



Absolute Pathnames

- The *pathnames* described in the previous slides start at the *root*.
- These *pathnames* are called "absolute pathnames".
- We can also talk about the *pathname* of a file *relative* to a directory.

Relative Pathnames

- If we are *in* the directory /home/hollid2, the relative pathname of the file **Syllabus** (inside "unix" dir) is:

unix/Syllabus

- Most unix commands deal with pathnames!
- We will usually use relative pathnames when specifying files.

The *current* directory and *parent* directory

- There is a special relative pathname for the current directory:

  .

- There is a special relative pathname for the parent directory:

  ..

- pwd: command to print working directory

Moving Around in the Filesystem

- cd (***c****hange* ***d****irectory*) command changes the current working directory
- The general form is: **cd [directoryname]**
- With no parameter, the current directory changes to your home directory.
- **cd** accepts a relative or absolute pathname:
  cd .. or **cd /home**

The `ls` command

- The `ls` command displays the names of some files (the default behavior doesn't display hidden files)
- If you give it the name of a directory as a *command line parameter* it will list all the files in the named directory
- The names of the files are shown (displayed) as relative pathnames

Example: the ls command

Try this:

```
ls       list files in current directory
ls /     list files in the root directory
ls .     list files in the current directory
ls ..    list files in the parent directory
ls /usr  list files in the directory /usr
```

ls command line options

- We can modify the output format of the ls program with a *command line option*

- To use a command line option precede the option letter with a minus:

ls -a or **ls -l**

- You can use 2 or more options at the same time like this:

ls -al

ls command line options

The ls command supports a bunch of options. E.g.:

- -l *long* format (include file times, owner, permissions)
- -a *all* (shows hidden* files as well as regular files)
- -A omit "." and ".." entries
- -F include special char to indicate file types.
- -1 list one file per line
- -R list everything in a directory and in all the subdirectories recursively (the entire hierarchy).
- And others more...

*hidden files have names that start with "."

More about commands

- You might want to know that Ctrl-C will cancel a command (stop the command)
- The brackets around [OPTION] and [FILE] in the general form of a command means that something is optional.
- We will see the general form of many commands described in this manner.
- Some commands have required parameters.
- Many commands support the --help option to display documentation of their syntax and valid options.
- man command is other form of displaying command's documentation

Copying Files

- The **cp** command copies files:

  cp [options] source dest

- The source is the name of the file you want to copy

- dest is the name of the new file

- source and dest can be relative or absolute

Another form of cp

- If destdir is a directory, cp will put a copy of source in the directory.

cp [options] source destdir

Yet another form of cp

- If you specify more than two names, cp assumes you are using this form:

cp [options] source... destdir

- In this case **cp** will copy multiple files to destdir.
- **source...** means at least one name (could be more than one)

Some Exercises

- Try giving **cp** three file names when the third is *not* a directory.
- Try to copy a directory with **cp**.
- Look at the man page for **cp**:

  man cp

Deleting (removing) Files

- The **rm** (remove) command deletes files:

rm [options] names...

- You can remove many files at once:

rm foo /tmp/blah /users/clinton/intern

rm Exercises

- Try to delete /etc/passwd
- Try to delete a directory
- Look at the man page for **rm**:

  man rm

mv (move)

- Moves files or directories from one place to another.

```
mv source1,source2 dest
```

- Also renames files (e.g. rename file.txt to test.txt):

```
mv file.txt test.txt
```

File attributes

- Every file (including dirs) has some attributes:
  - Access Times:
    - when the file was created
    - when the file was last changed
    - when the file was last read
  - Size
  - Owners (user and group)
  - Permissions

File Owners

- Each file is owned by a user
- You can find out the username of the file's owner using "ls -l".
- Each file is also owned by a Unix group.
- **ls -l** also shows the group that owns the file.

File Permissions

- Each file has a set of permissions that control who can mess with the file.

- There are three kinds of permissions:
  - read abbreviated “r”
  - Write abbreviated “w”
  - execute abbreviated “x”

- There are separate permissions for the file owner, group owner and everyone else.

ls -l

```
> ls -l foo
-rw-rw----   1 hollingd grads 13 Jan 10 23:05 foo
```

permissions

owner

group

size

last modified date

file name

-rwxrwxrwx

Owner Group Others

Type of file:

- **- means plain file**
- **d means directory**

rwx

- Files:
  - **r** - allowed to read
  - **w** - allowed to write
  - **x** - allowed to execute
- Directories:
  - **r** - allowed to see the names of the file
  - **w** - allowed to add and remove files
  - **x** - allowed to enter the directory

Changing Permissions

- The **chmod** command changes the permissions associated with a file or directory.
- There are a number of forms of chmod, this is the simplest:

chmod mode file

chmod

- "mode" has the following form*:

  [ugoa][+-=][rwx]

u=user g=group o=other a=all

\+ add permission

\- remove permission

= set permission

*The form is really more complicated, but this simple version will do enough for now.

chmod examples

```
> ls -al foo
rwxrwx--x   1 hollingd grads …

> chmod o-x foo
> ls -al foo
rwxrwx---   1 hollingd grads

> chmod u-r .
> ls -al foo
ls: .: Permission denied
```

Other filesystem and file commands

- **mkdir** make directory
- **du**: directory size
- **stat** displays the detailed status of a particular file or a file system
- **rmdir** remove directory
- **touch** change file timestamp (or create a blank file)
- **cat** concatenate files and print out to terminal