

Unix Accounts and the Filesystem

Unix Accounts

- To access a Unix system you need to have an *account*.
- Unix account includes:
 - username and password
 - userid and groupid
 - home directory
 - shell

username

- A username is (typically) a sequence of alphanumeric characters of length no more than 8.
- username the primary identifying attribute of your account.
- username is (usually) used as an email address
- the name of your home directory is usually related to your username.

password

- a password is a secret string that only the user knows (not even the system knows!)
- When you enter your password the system encrypts it and compares to a stored string.
- passwords are (usually) no more than 8 characters long.
- It's a good idea to include numbers and/or special characters (don't use an english word!)

userid

- a userid is a number (an integer) that identifies a Unix account. Each userid is unique.
- It's easier (and more efficient) for the system to use a number than a string like the username.
- You don't need to know your userid!

Unix Groups and groupid

- Unix includes the notion of a "group" of users.
- A Unix group can share files and active processes.
- Each account is assigned a "primary" group.
- The groupid is a number that corresponds to this primary group.
- A single account can belong to many groups (but has only one primary group).

Home Directory

- A home directory is a place in the file system where the account files are stored.
- 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

- A Shell is a unix program that provides an interactive session - a text-based user interface.
- When you log in to a Unix system the program you initially interact with is your shell.
- There are a number of popular shells that are available.

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!

Session Startup

- Once you log in, your shell will be started and it will display a prompt.
- When the shell is started it looks in your home directory for some customization files.
 - You can change the shell prompt and a bunch of other things by creating customization files (more on this later...)

Your Home Directory

- Every Unix process* has a notion of the “current working directory”.
- Your shell (which is a process) starts with the current working directory set to your home directory.

*A process is an instance of a *program* that is currently running.

Interacting with the Shell

- The shell prints a prompt and waits for you to type in a command.
- The shell can deal with a couple of types of commands:
 - shell internals - commands that the shell handles directly.
 - External programs - the shell runs a program for you.

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

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!
 - how long depends on your specific flavor of Unix

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.

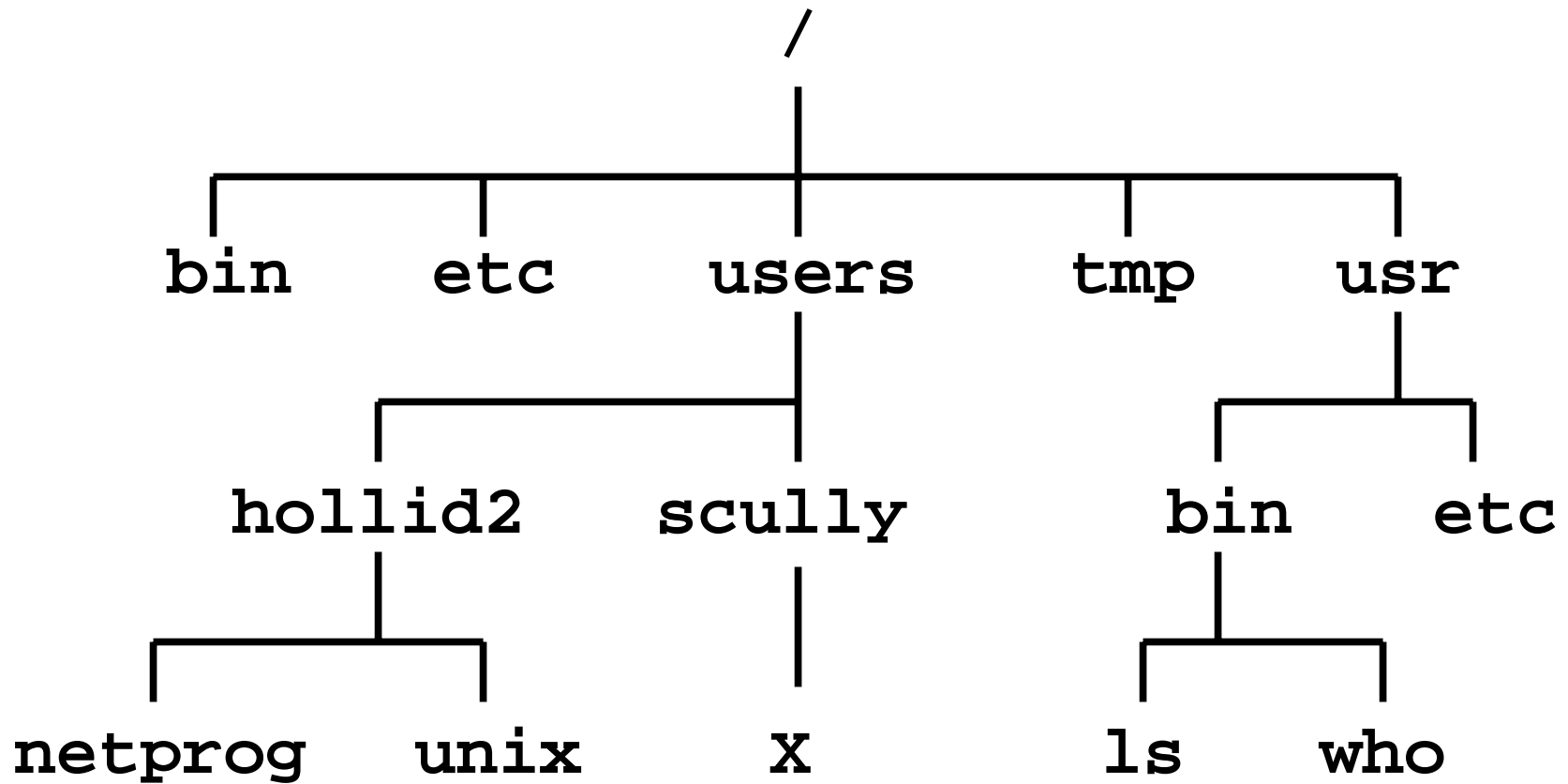
Directories

- 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).
- Mac and Windows weenies*: A directory is the same idea as a *folder*.
- *weenies is actually a term usually used to describe Unix users - I'm being defensive...

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.

The Filesystem



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 /

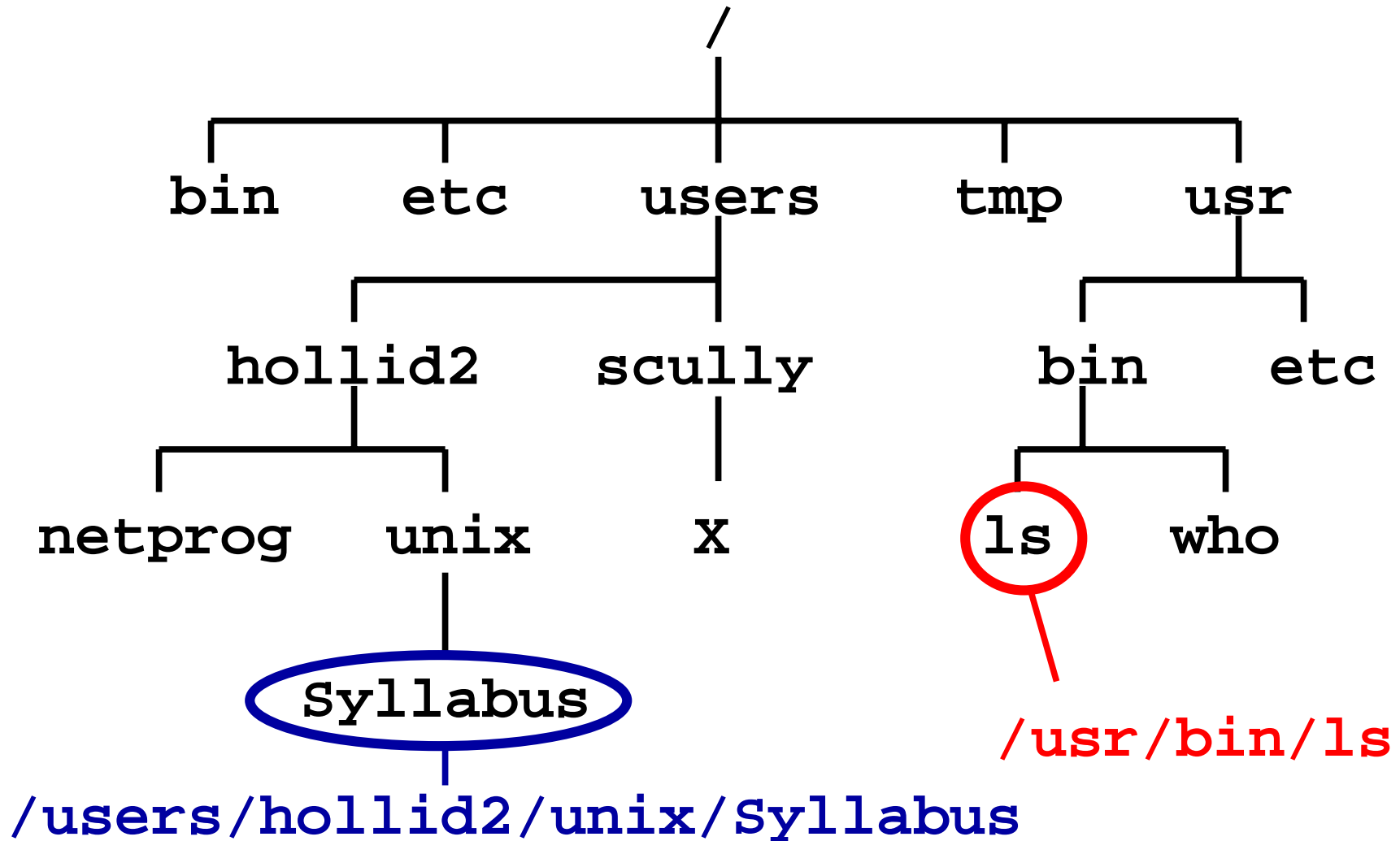
Pathnames

- The *pathname* of a file includes the file name and the name of the directory that holds the file, and the name of the directory that holds the directory that holds the file, and the name of the ... 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 `/users/hollid2`, the relative pathname of the file **Syllabus** is:

unix/Syllabus

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

Example: The ls command

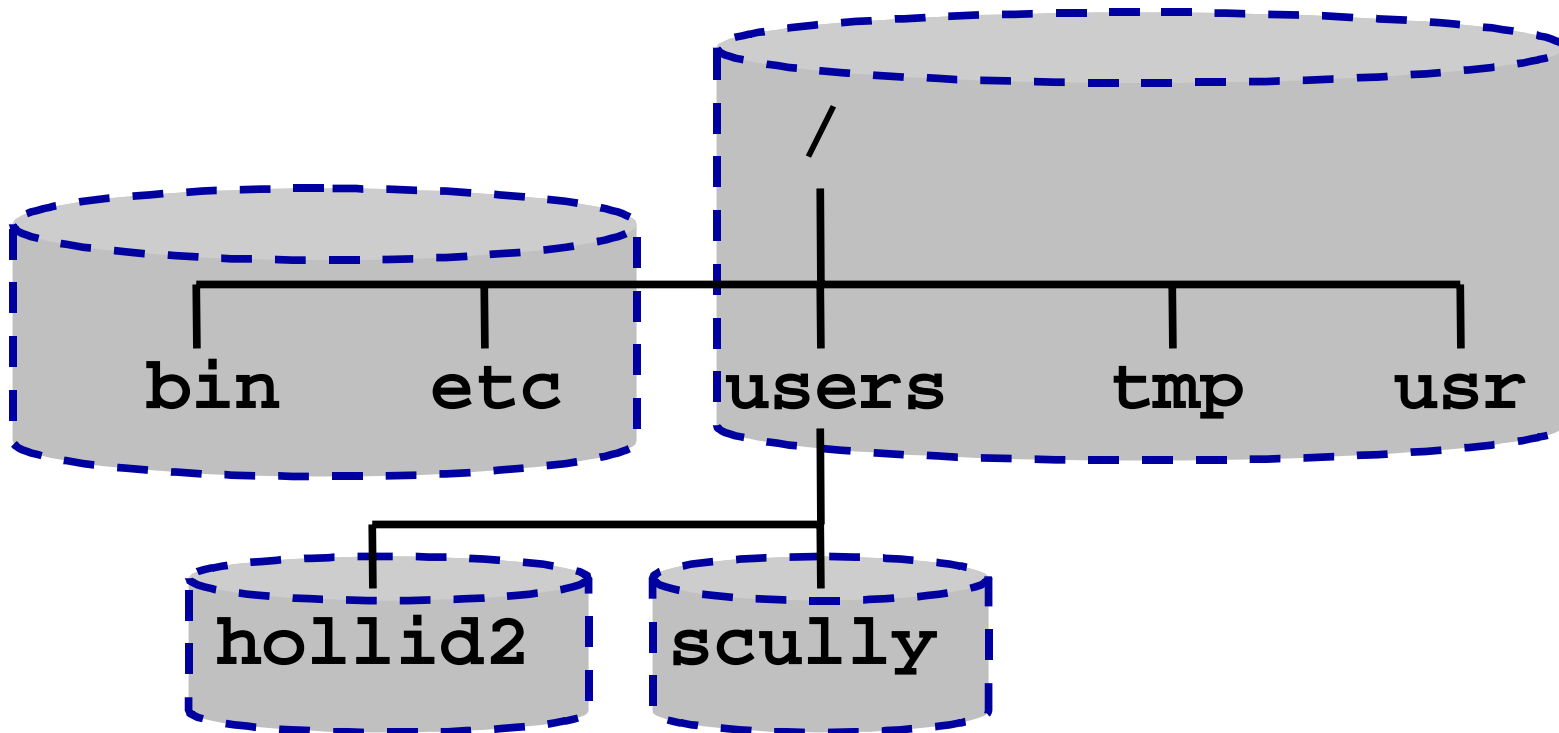
- Exercise: login to a unix account and type the command "ls".
- The names of the files are shown (displayed) as relative pathnames.
- Try this:

```
ls /usr
```

- `ls` should display the name of each file in the directory `/usr`.

Disk vs. Filesystem

- The entire hierarchy can actually include many disk drives.
 - some directories can be on other computers



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:

••

The ls command

- The ls command displays the names of some 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.

Some things to try

`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`

Command Line Options

- We can modify the output format of the ls program with a *command line option*.
- The ls command support a bunch of options:
 - l *long* format (include file times, owner and permissions)
 - a *all* (shows hidden* files as well as regular files)
 - F include special char to indicate file types.

*hidden files have names that start with "."

ls command line options

- 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
```

General `ls` command line

- The general form for the `ls` command is:

```
ls [options] [names]
```

- The options must come first!
- You can mix any options with any names.
- An example:

```
ls -al /usr/bin
```

ls [options] [names]

- The brackets around options and names in the general form of the ls 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 names

- You can give the ls command many names:

```
ls /usr /etc
```

```
ls -l /usr/bin /tmp /etc
```

Moving Around in the Filesystem

- There cd command can change the current working directory:

- **cd** *change directory*

- The general form is:

cd [directoryname]

cd

- With no parameter, the `cd` command changes the current directory to your home directory.
- You can also give `cd` a relative or absolute pathname:

```
cd /usr
```

```
cd ..
```

Some more commands and command line options

- **ls -R** will list everything in a directory and in all the subdirectories recursively (the entire hierarchy).
 - you might want to know that Ctrl-C will cancel a command (stop the command)!
- **pwd**: print working directory
- **df**: shows what disk holds a directory.

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 you specify a dest that is a directory, `cp` will put a copy of the source in the directory.
- The filename will be the same as the filename of the source file.

```
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` command deletes files:

```
rm [options] names...
```

- `rm` stands for "remove".
- 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

File attributes

- Every file 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 Time Attributes

- Time Attributes:
 - when the file was last changed `ls -l`
 - when the file was created* `ls -lc`
 - when the file was last read (accessed) `ls -ul`

*actually it's the time the file status last changed.

File Owners

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

File Permissions and RCS

- NOTE: The description of file permissions that follows is for general Unix systems.
- RCS uses a special kind of filesystem that has a much more extensive way of handling file permissions!
- We will look at AFS (the file system used by RCS) later.

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 hollind grads 13 Jan 10 23:05 foo
```

permissions owner group size time name

ls -l and permissions

-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 file

- 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 hollind grads ..
```

```
> chmod g-wx foo  
> ls -al foo  
-rwxrw----    1 hollind grads
```

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

Other filesystem and file commands

- **mkdir** make directory
- **rmdir** remove directory
- **touch** change file timestamp (can also create a blank file)
- **cat** concatenate files and print out to terminal.