What is UNIX?

• UNIX is a time-sharing operating system with user-chosen shells (user interfaces) and one kernel (operating system core, which allocates and control resources such as CPU time, memory and I/O devices).

• UNIX includes:
  – kernel
  – tools, including compilers, editors, etc.
  – shell, which includes its own programming language
What is LINUX?

- Linux is an open-source operating system, indirectly based on the last public release of UNIX.
- Linux is available in many different versions and different releases and is also closely associated with the GNU project, and through the GNU project has many tools comparable to those found in a UNIX distribution.

History of UNIX

- UNIX was developed at AT&T Bell Labs and is one of the first time-sharing operating systems.
- It was originally developed on a DEC PDP-7 and later redeveloped on a DEC PDP-11 in C, the first operating system written in a high-level language.
- It became popular at colleges and universities and later in the financial industry.
History of Linux

- Andrew Tanenbaum developed MINIX from the last public distribution of UNIX for use with his operating Systems textbook.
- When version 2 of MINIX was released, it was not well adapted for 32-bit processors. This inspired Linus Torsvald to begin work on what became Linux.
- Torsvald welcomed suggestions; this gave way to the community approach to software development that became a hallmark of Linux.

UNIX and Terminals

- Unix is **full duplex** – communications between computer and terminal goes in both directions simultaneously and the computer controls terminal display, using a process called `echo`.
- Example
  
  ```
  echo hi there
  hi there
  ```
Control Characters

- Control characters serve a special purpose, performing tasks that one character cannot normally do.
- These include:

<table>
<thead>
<tr>
<th>Control Character</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>RETURN (^m)</td>
<td>signifies end of line</td>
</tr>
<tr>
<td>^d</td>
<td>EOF (end of file)</td>
</tr>
<tr>
<td>^g</td>
<td>rings the bell</td>
</tr>
<tr>
<td>^h</td>
<td>backspace</td>
</tr>
<tr>
<td>^i</td>
<td>tab</td>
</tr>
<tr>
<td>DELETE</td>
<td>deletes the character to which the cursor points</td>
</tr>
<tr>
<td>BREAK</td>
<td>stops a program immediately</td>
</tr>
</tbody>
</table>

Logging In

login: siegfrie
password: <enter> doesn't appear when you type
login as: siegfrie
siegfrie@panther.adelphi.edu's password:

Welcome to Panther!
If you experience problems or have questions, please contact CustomerServices at 516.877.3340, email support@adelphi.edu, or visit us in the Information Commons on the Second Floor of Swirbul Library.

** Reminder: Your Website is
http://home.adelphi.edu/~siegfried

[SIEGFRIE@panther ~]$
Commands to Try

- **date** – gives date and time
- **who** – tells you who is on the system and connected to what terminal
- **who am i** – tells you who you are and to what terminal you are connected
- **whoami** – tells you who you are
- **w** – displays what the system users are doing

**stty**

- UNIX gives the user a way of adjusting terminal settings
- **stty** – set terminal – change and print terminal line settings.
- Example
  
  
  [SIEGFRIE@panther ~] $ stty -all
  
  speed 38400 baud; rows 24; columns 80; line 0;
  intr = ^C; quit = ^\; erase = ^?; kill = ^U; eof
  = ^D; eol = <undef>; undefined
  
  ... ...
## Standard Codes

<table>
<thead>
<tr>
<th>Code</th>
<th>Command</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td><code>intr</code></td>
<td><code>^c</code></td>
<td>Stops a program</td>
</tr>
<tr>
<td><code>erase</code></td>
<td><code>^h</code></td>
<td>Backspaces</td>
</tr>
<tr>
<td><code>werase</code></td>
<td><code>^w</code></td>
<td>Erases last word typed</td>
</tr>
<tr>
<td><code>kill</code></td>
<td><code>^u</code></td>
<td>Kills the current input line</td>
</tr>
<tr>
<td><code>quit</code></td>
<td><code>^\</code></td>
<td>Stops the program and saves core in a file</td>
</tr>
<tr>
<td><code>stop</code></td>
<td><code>^s</code></td>
<td>Pauses screen output</td>
</tr>
<tr>
<td><code>start</code></td>
<td><code>^q</code></td>
<td>Resumes screen output</td>
</tr>
<tr>
<td><code>eof</code></td>
<td><code>^d</code></td>
<td>No more data</td>
</tr>
<tr>
<td><code>suspend</code></td>
<td><code>^z</code></td>
<td>Temporarily stops (i.e., suspends) a program</td>
</tr>
<tr>
<td><code>resume</code></td>
<td><code>^y</code></td>
<td>Resumes running a program</td>
</tr>
</tbody>
</table>

## Type-ahead

- You can type as far ahead as you wish as long as you don't exceed the keyboard buffer's storage capacity.
Stopping A Program

• Programs can be stopped by pressing the Break, usually the ^c.
• ^z stops a program temporarily. You can restart it by pressing ^y on most systems.

  Don't logout without terminating suspended programs!

Logging Out

• You can log out by typing logout or exit. If you use the wrong one, the system will let you know.
• On some older systems, you can logout by simply typing ^d.
write

$write SIEGFRIE

Message from SIEGFRIE@panther.adelphi.edu on pts/0 at 13:54 ...
this is a test
this is a test
^d
EOF
[SIEGFRIE@panther ~]$%

Online manual

• Most UNIX (and Linux) systems have an online manual that can be accessed by typing:
  \texttt{man\ commandname}

• Example
  \texttt{man who}\hspace{1em}– shows the manual page on \texttt{who}
  \texttt{man man}\hspace{1em}– shows the manual page on \texttt{man}
Files and File-Oriented Commands

- A great deal of work on the system involves files (data moving into or out of the computer), which makes file-oriented commands particularly important.
- File commands include:
  - `vi`, `ex`, `ed` – file editors
  - `cat`, `pr` – printing and display commands
  - `mv`, `cp`, `rm` – file manipulation commands
  - `grep`, `sort`, `diff`, `tail` - filters

Text editors

- There are 3 text editors that we will concern ourselves with:
  - `ed` – the original line-oriented editor
  - `ex` – the extended line-oriented editor
  - `vi` – visual editor (screen-oriented)
A Sample **ed** Session

```
$ ed
a  add text
...
type lots of stuff
...
.  ends adding text
w junk  write text to a file called junk
39  number of characters saved
q  quit
```

Changing A File

```
[SIEGFRIE@panther ~]$ ed junk
junk: No such file or directory
a
To be or not to be
.
a
That is the question
.
p
That is the question
l,$p
To be or not to be
That is the question
```
Changing A File (continued)

s/the/The/
l,$p

To be or not to be
That is The question

w
40
q
[SIEGFRIE@panther ~]$
ls – l

- ls -l – provides a long listing of the files

[SIEGFRIE@panther bbb]$ ls -l long listing
total 8
-rw-r--r-- 1 SIEGFRIE users 40 Jun 9 16:49 junk
-rw-r--r-- 1 SIEGFRIE users 40 Jun 9 17:03 temp

Permissions
# of links to file
owner
owner's group
# of bytes in file
date & time of last modification
file name

Permission

Owner has read and write but not execute permission
Owner's group has read but not write nor execute permission
The rest of the world has read but not write nor execute permission
**cat** – Displaying a File

[SIEGFRIE@panther bbb]$ cat temp
To be or not to be
[SIEGFRIE@panther bbb]$ cat junk
That is The question
[SIEGFRIE@panther bbb]$ cat temp junk
To be or not to be
That is The question
[SIEGFRIE@panther bbb]$

---

**pr**

- **pr** displays files in format suitable for printing.
- **pr**  
  - *n* display the file in *n*-column format for printing.

- Different systems have different commands for printer access. **lp** is a very common command.
**mv, cp and rm**

- **mv** – move (or rename) a file
- **cp** – copy a file
- **rm** – remove (or delete) a file

**mv, cp and rm – An Example**

```bash
[SIEGFRIE@panther bbb]$ ls
junk temp
[SIEGFRIE@panther bbb]$ mv junk precious
[SIEGFRIE@panther bbb]$ ls
precious temp
[SIEGFRIE@panther bbb]$ cp precious junk
[SIEGFRIE@panther bbb]$ ls
junk precious temp
[SIEGFRIE@panther bbb]$ rm precious
[SIEGFRIE@panther bbb]$ ls
junk temp
[SIEGFRIE@panther bbb]$
```
Rules Governing File Names

- Older systems may limit names to 14 characters.
- UNIX is case-sensitive; junk, Junk and JUNK are three different files.
- File names should not (but unfortunately can) include unprintable characters (like escape characters) or characters with special meanings.
  - E.g., how would you print the file –t?

A Few Helpful File Processing Commands

- There are several file processing commands that will become useful:

<table>
<thead>
<tr>
<th>Command</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>wc</td>
<td>word count</td>
</tr>
<tr>
<td>grep</td>
<td>general regular expression program – recognizes text within a file.</td>
</tr>
<tr>
<td>sort</td>
<td>sorts lines of text within a file.</td>
</tr>
<tr>
<td>tail</td>
<td>prints the last line(s) of text within a file.</td>
</tr>
<tr>
<td>cmp</td>
<td>compares two files, printing the first pair of lines that differ.</td>
</tr>
<tr>
<td>diff</td>
<td>compares two files, printing each pair of lines that differ.</td>
</tr>
</tbody>
</table>
Great fleas have little fleas
upon their backs to bite 'em
And little fleas have lesser fleas
and so on ad infinitum.
And the great fleas themselves, in turn,
have greater fleas to go on;
While these again have greater still,
and great still and so on.

And little fleas have lesser fleas,
and so on ad infinitum.
grep

[SIEGFRIE@panther bbb]$ grep fleas poem
Great fleas have little fleas
And little fleas have lesser fleas,
And the great fleas themselves, in turn,
  have greater fleas to go on;

[SIEGFRIE@panther bbb]$ grep -v fleas poem
upon their backs to bite 'em
and so on ad infinitum.
While these again have greater still,
  and great still and so on.

sort

[SIEGFRIE@panther bbb]$ sort poem
  and great still and so on.
And little fleas have lesser fleas,
  and so on ad infinitum.
And the great fleas themselves, in turn,
Great fleas have little fleas
  have greater fleas to go on;
```
cmp

[SIEGFRIE@panther bbb]$ cat newpoem
Great fleas have little fleas
upon their backs to bite them
And little fleas have lesser fleas,
and so til ad infinitum.
And the great fleas themselves, in turn,
have greater fleas to go on;
While these again have greater still,
and great still and so on.

[SIEGFRIE@panther bbb]$ cmp poem newpoem
poem newpoem differ: byte 57, line 2
```

tail

```
[SIEGFRIE@panther bbb]$ tail -1 poem
and great still and so on.
[SIEGFRIE@panther bbb]$ tail +3 poem
And little fleas have lesser fleas,
and so on ad infinitum.
And the great fleas themselves, in turn,
have greater fleas to go on;
While these again have greater still,
and great still and so on.
[SIEGFRIE@panther bbb]$
```
diff

[SIEGFRIE@panther bbb]$ diff poem newpoem
2c2
<  upon their backs to bite 'em
---
>  upon their backs to bite them
4c4
<  and so on ad infinitum.
---
>  and so til ad infinitum.
[SIEGFRIE@panther bbb]$

Command Summary

<table>
<thead>
<tr>
<th>Command</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td><code>ls</code></td>
<td>list filenames in current directory</td>
</tr>
<tr>
<td><code>ls filenames</code></td>
<td>lists only these files</td>
</tr>
<tr>
<td><code>ls -t</code></td>
<td>lists in reverse chronological order</td>
</tr>
<tr>
<td><code>ls -l</code></td>
<td>long listing</td>
</tr>
<tr>
<td><code>ls -u</code></td>
<td>list by last time used</td>
</tr>
<tr>
<td><code>ls -r</code></td>
<td>list in reverse order</td>
</tr>
<tr>
<td><code>ed filename</code></td>
<td>edit a file listed by name</td>
</tr>
<tr>
<td><code>cp file1 file2</code></td>
<td>copy file1 to file2</td>
</tr>
<tr>
<td><code>mv file1 file2</code></td>
<td>move (or rename) file1 to file2</td>
</tr>
<tr>
<td><code>rm filenames</code></td>
<td>delete these files</td>
</tr>
</tbody>
</table>
## Command Summary

<table>
<thead>
<tr>
<th>Command</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td><code>cat filename(s)</code></td>
<td>display file contents</td>
</tr>
<tr>
<td><code>pr filename(s)</code></td>
<td>display and format file contents</td>
</tr>
<tr>
<td><code>pr -n filename(s)</code></td>
<td>display and format file contents in n columns</td>
</tr>
<tr>
<td><code>pr -m filename(s)</code></td>
<td>display files side by side</td>
</tr>
<tr>
<td><code>wc filename(s)</code></td>
<td>count words and bytes for these files</td>
</tr>
<tr>
<td><code>wc -l filename(s)</code></td>
<td>count lines for these files</td>
</tr>
<tr>
<td><code>grep pattern filename(s)</code></td>
<td>print lines containing the pattern</td>
</tr>
<tr>
<td><code>grep -v pattern filename(s)</code></td>
<td>print lines not containing the pattern</td>
</tr>
<tr>
<td><code>cmp file1 file2</code></td>
<td>print line of first difference</td>
</tr>
<tr>
<td><code>diff file1 file2</code></td>
<td>print each pair of differing lines</td>
</tr>
</tbody>
</table>

## Directories

- The system knows how to distinguish between different files with the same name by recognizing that they are listed in **different directories**.
- Each user has his/her own directory, which can be divided into subdirectories by the user.
- You can use the `cd` (change directory) command to switch between different directories and `pwd` (print working directory) to display the current directory being used.
cd and pwd – An Example

```
[SIEGFRIE@panther ~]$ cd bbb
[SIEGFRIE@panther bbb]$ ls
junk newpoem poem temp
[SIEGFRIE@panther bbb]$ pwd
/home/siegfried/bbb
[SIEGFRIE@panther bbb]$ ls /
bin     dev initrd     media opt   sbin
        ... ...
delte_this home lost+found mnt root srv
[SIEGFRIE@panther bbb]$ cd
[SIEGFRIE@panther ~]$ pwd
/home/siegfried
[SIEGFRIE@panther ~]$ ls
a.out                           compethics.doc     j
args                            concord.cpp.txtcs343  java
        ... ...
calendar                         HW2.doc
```

Working With Subdirectories

```
[SIEGFRIE@panther bbb]$ cat /home/siegfried/bbb/junk
That is The question
[SIEGFRIE@panther bbb]$ cat junk
That is The question
[SIEGFRIE@panther bbb]$ cd ..
[SIEGFRIE@panther ~]$ pwd
/home/siegfried
[SIEGFRIE@panther ~]$ cp bbb/junk junk
[SIEGFRIE@panther ~]$ cat junk
That is The question
[SIEGFRIE@panther ~]$ 
```
A Sample File System

```
/m
  └── bin
  └── dev
  └── etc
  └── tmp
  └── unix
  └── boot
     └── you
     └── mike
     └── paul
     └── mary
        └── junk
        └── junk
        └── temp
        └── junk
        └── data
```

More Work With Subdirectories

```
[SIEGFRIE@panther home]$ ls /bin
alsaunmute     date           grep      mkdir
        ... ...
ggettext        mailx          rm          tracepath
[SIEGFRIE@panther home]$ ls /usr/bin
[ 4odb
  4rdf
  4ss
        ... ...
ktip
ktradertest    znew
[SIEGFRIE@panther home]$ /bin/date
Wed Jun 10 10:40:44 EDT 2009
[SIEGFRIE@panther home]$ /bin/who
-bash: /bin/who: No such file or directory
[SIEGFRIE@panther home]$
```
Creating and Deleting Subdirectories

[SIEGFRIE@panther bbb]$ mkdir book
[SIEGFRIE@panther bbb]$ cd book
[SIEGFRIE@panther book]$ pwd
/home/siegfried/bbb/book
[SIEGFRIE@panther book]$ cd
[SIEGFRIE@panther ~]$ pwd
/home/siegfried
[SIEGFRIE@panther ~]$ ls bbb
book junk newpoem poem temp
[SIEGFRIE@panther ~]$ rmdir bbb/book
[SIEGFRIE@panther ~]$ ls bbb
junk newpoem poem temp
[SIEGFRIE@panther ~]$ 

The Shell

- The shell is another name for the command interpreter, which runs all the commands that we type in a session on a UNIX (or Linux) system.
- It provides 3 benefits:
  - File name shorthands - a whole bunch of files specified using wild card characters, can be specified at one.
  - Input-output redirection – a file can replace either keyboard or screen or both.
  - You can personalize your environment.
Shorthands

• Imagine that every chapter section is a separate file ch1.1, ch1.2, ch1.3, … ch2.1, ch2.2

We can print them by typing

pr ch1.1 ch1.2 ch1.3 ... ch2.1 ch2.2

or

pr ch*

• wc ch1.* gives us a count of characters, words and lines for all the sections of chapter 1.

Shorthands – An Example

[SIEGFRIE@panther bbb]$ wc ch*
1 421 ch1.1
8 260 47 ch1.2
8 258 47 ch2.1
1 19 6 ch2.2
18 558 104 total

[SIEGFRIE@panther bbb]$
echo

- **echo** – displays a message on the screen.
  
  ```
  [SIEGFRIE@panther bbb]$ echo hello world
  hello world
  [SIEGFRIE@panther bbb]$ echo ch1.*
  ch1.1 ch1.2
  [SIEGFRIE@panther bbb]$ echo *
  ch1.1 ch1.2 ch2.1 ch2.2
  [SIEGFRIE@panther bbb]$
  ```

- **pr** * - displays all the files in print format.
- **rm** * - deletes all files in current directory.
- **rm** *.sav – deletes all files ending with *.sav
• [ ] matches a single occurrence of one of the characters in the brackets.
• `pr ch[12346789]` prints every whole chapter except 5.
• `pr ch[1-46-9]` prints every whole chapter except 5.
• `rm temp[a-z]` – deletes `tempa, tempb, ..., tempz` if they exist.

• ? – replaces any single occurrence of a single character.
• `ls ?` – lists single-character file names.
• `ls -l ch?.1` lists `ch1.1, ch2.1, ... ch9.1` if they exist.
• `rm temp?` – deletes `temp1, temp2, ...tempa, tempb, ..., tempz` if they exist.
Final Word on Metacharacters

• All file names must exist for the metacharacters to be used:
  \texttt{mv \ ch.* \ chapter.*}
  Won’t work because the \texttt{chapter} files don’t already exist.

• Metacharacters also can match other names in the path, e.g., \texttt{usr/*/calendar}.

• How do you use a file name that has a metacharacter in it?
  • \texttt{ls '?>} or \texttt{ls \\?

Redirection

• Redirection replaces standard input (or standard output) with a file.
  • \texttt{ls} – lists the files in the directory on the screen.
  • \texttt{ls > filelist} – creates a file containing the directory listing
  • \texttt{cat f1 f2 f3 >temp} – places the files' contents in a file called \texttt{temp}. If the file already exists, it is overwritten.
  • \texttt{cat f1 f2 f3 >temp} – places the files' contents in a file called \texttt{temp}. If the file already exists, the output is placed at the end.
Redirection (continued)

- **mail mary \(\downarrow\)**
  
  You have to type the mail message.

- **mail mary <message**
  
  It will mail the contents of the file `message`.

---

**Redirection – Some Other Examples**

- `who > temp`
- `sort < temp`

- `who > temp`
  `wc -l < temp`

- `ls > temp`
  `pr -3 < temp`

- `who > temp`
  `grep mary < temp`

- `sort < temp`
- `sort temp`

- Alphabetical list of users
- Counts number of users
- Prints filenames in 3-column format
- Is Mary logged in?
- Does the same thing
Using `sort` Without A File

```
[SIEGFRIE@panther bbb]$ sort
def
ijk
abc
^d
abc
def
ijk
[SIEGFRIE@panther bbb]$
```

Pipes

- We don’t need to use temporary files the way we did before.
  
  ```
  who | sort
  who | wc -l
  ls | wc -l
  ls | pr -3
  who | grep mary
  ```

- Any program reading the keyboard and displaying on the screen can use a pipe (|)
  
  ```
  who | grep mary | wc -l
  ```

*How many times did Mary log in?*
Normal Flow of Data

Processes

- A process is the act of running a program
- Proper use of the shell will allow you to run 2 processes with one command:

```
[SIEGFRIE@panther bbb]$ date;who
[SIEGFRIE@panther bbb]$
```

```
SIEGFRIE pts/0 Jun 10 10:07 (pool... ... .net)
[SIEGFRIE@panther bbb]$
```
Background Processes

- Proper use of the shell will allow you to run 2 processes concurrently:

```bash
[SIEGFRIE@panther bbb]$ wc ch* > wc.out &
[1] 12909
[SIEGFRIE@panther bbb]$ cat wc.out
1 4 21 ch1.1
8 47 260 ch1.2
8 47 258 ch2.1
1 6 19 ch2.2
18 104 558 total
[1]+  Done    wc ch* >wc.out
[SIEGFRIE@panther bbb]$
```

Pipes and Background Processes

```
pr ch* | lp &
6951
```

```
wait
kill -9 6944
```

- Applies to the whole pipe
- Waits until all the processes are finished
- Kills the lp process
ps

• ps – process status

[SIEGFRIE@panther bbb]$ ps ag

<table>
<thead>
<tr>
<th>PID</th>
<th>TTY</th>
<th>STAT</th>
<th>TIME</th>
<th>COMMAND</th>
</tr>
</thead>
<tbody>
<tr>
<td>6660</td>
<td>tty1</td>
<td>Ss+</td>
<td>0:00</td>
<td>/sbin/mingetty tty1</td>
</tr>
<tr>
<td>6661</td>
<td>tty2</td>
<td>Ss+</td>
<td>0:00</td>
<td>/sbin/mingetty tty2</td>
</tr>
<tr>
<td>6662</td>
<td>tty3</td>
<td>Ss+</td>
<td>0:00</td>
<td>/sbin/mingetty tty3</td>
</tr>
<tr>
<td>6663</td>
<td>tty4</td>
<td>Ss+</td>
<td>0:00</td>
<td>/sbin/mingetty tty4</td>
</tr>
<tr>
<td>6664</td>
<td>tty5</td>
<td>Ss+</td>
<td>0:00</td>
<td>/sbin/mingetty tty5</td>
</tr>
<tr>
<td>6665</td>
<td>tty6</td>
<td>Ss+</td>
<td>0:00</td>
<td>/sbin/mingetty tty6</td>
</tr>
<tr>
<td>12115</td>
<td>pts/0</td>
<td>Ss</td>
<td>0:00</td>
<td>-bash</td>
</tr>
<tr>
<td>12944</td>
<td>pts/0</td>
<td>R+</td>
<td>0:00</td>
<td>ps ag</td>
</tr>
</tbody>
</table>

[SIEGFRIE@panther bbb]$

Creating Processes

• Processes can create processes.

• If a process is running, in the background when you log out, it will terminate unless you wrote nohup (no hang-up).

  nohup command &

• Output is saved in the file nohup.out

  nice expensive-command – cuts the process's priority