Recitation 5
Outline

• Terminals
• Shells
• How does “ls” work?
• Linux Concepts
• How does “ls” work?
Terminals
What is a terminal?
Physical Terminals

Electromechanical terminal

Televideo ASCII character mode terminal
Physical Terminal Interaction

Host Computer

Terminal

Keyboard

Display

Serial port

Program
Virtual Terminals

**SSH Secure Shell**

![SSH Secure Shell]

```

Welcome to the Carnegie Mellon University
Red Hat Linux timeshare pool

These machines are a shared resource. Please be considerate of other users.
[Image: http://www.cmu.edu/computing/documentation/policies_servers/servers.html]

[jprimer0@unix35 ~]
```

**Putty**

![Putty]

```
jprimer0@unix35 ~]
$ ls
install_cisco_vpn.exe
network cisco-vpn-0_9_14_001.zip
Desktop poster.png
Lab_one_1.pdf private
[Cherry] private
/home/ linux public
/home/private rosters
asst.pdf send.py
asst.txt asusted-dream_devphone_userdebug-im9-15027.zip
 Ask svn
ask.tgs things
ask.tmp
bith.c tmp.c
lablab connection thlab-writeup.pdf
cheatsheet.pdf wireless_vgm.htm
class uwo

[jprimer0@unix35 ~]
```

**xterm**

![xterm]

```
[jprimer0@l0dp4x9 ~]
$ date
Sun Mar 12 12:58:02 PST 2006

[jprimer0@l0dp4x9 ~]
$ fortune
I stopped believing in Santa Claus when I was six. Mother took me to
see him in a department store and he asked for my autograph.

-- Shirley Temple

[jprimer0@l0dp4x9 ~]
$ date
```

---
Virtual Terminal Interaction
SSH Terminal Interaction

Host Computer
- VTerminal
- VTerminal
- VTerminal

Remote Computer
- Program

Network

Keyboard

Display
SSH Terminal Interaction
Shells
What is a shell?

• You’ve all seen it (hopefully)
What is a shell?

- A shell is the most fundamental program for a terminal to run
- A shell allows a user to run and control multiple processes
What is a shell?

- Shells can be graphical as well
- You can think of it as the user interface to a particular operating system
How does “ls” work?

bash> ls
folder  file
bash>
Basic Idea

Terminal

Keyboard

Display

bash>

Shell
(1) “ls” is sent to the shell
(2) The shell creates an “ls” process
(3) The shell gives control of the terminal to “ls”
(4) “ls” writes to the display
(5) “ls” finishes, the shell then regains control of the terminal
(6) The shell prints the prompt and waits for next command
Ok so far, but something is missing...
How does shell actually interact with the terminal?
bash>ls
folder file
bash>
Since you are going to be interacting with the kernel you should be familiar with some Linux concepts...
Linux Concepts
TTY

• The use of terminals are tightly integrated with the Linux kernel
• These terminals are known are part of a Linux subsystem referred to as the TTY subsystem
TTY Subsystem

Terminal

- Keyboard
- Display

Kernel

TTY driver

Shell

bash> ls
folder file
bash>
Multiple Terminals

Kernel

TTY driver

Terminal

Keyboard

Display

Terminal

Keyboard

Display

Shell
Each terminal is referred to as tty#...
Each process is associated with a controlling terminal...
But how does a process refer to some specific terminal?
Linux Devices

• Linux Devices are computer devices which are controlled by the kernel
  – Mouse, Keyboard, Wi-fi chip, disks, and terminals...

• Each device is represented in Linux as a file usually located in /dev
Linux Devices

- Each device is represented in Linux as a file usually located in /dev
Linux Devices

- Reading or writing to these files allow processes to control these devices
What does reading from stdin and writing to stdout do?
But how does a process refer to some specific terminal?
Process Groups

• A collection of one or more processes
• Children created from a “fork” call by a parent automatically belong to the same process group
Changing Process Groups

• A process can change which process group it belongs to by using the system call setpgid()

• int setpgid(pid_t pid, pid_t pgid);
setpgid(243, 0);
Updated View

Kernel

TTY driver

tty1

Keyboard

Display

tty2

Keyboard

Display

Process Group

Shell: tty1
Foreground Process Group

- Every active terminal has some controlling process group referred to as the “Foreground Process Group”
- Processes in this group can read/write to the terminal and receive signals originating from this terminal
- All other process groups are “Background Process Groups” and cannot read/write to the terminal or receive signals
One Foreground Process Group Per TTY
Changing the Foreground Process Group

- A process can change the Foreground Process Group by using the system call `tcsetpgrp()`

- `int tcsetpgrp(int fildes, pid_t pgid);`
tcsetpgrp(STDIN_FILENO, 213);
How does “ls” work?

bash> ls
folder  file
bash>
More Advanced Idea
(1) "ls" is sent to the kernel and handled by the TTY driver
(2) The “Shell” does a read or fgets system call to get input from the terminal
(3) Since “Shell” is part of the Foreground Process group it is able to read from the terminal and receives the “ls” string.
(4) The shell processes “ls” and attempts to create a new child by using the fork system call.
(5) fork() returns twice, one to the shell process and the other to the newly created child process
(6) Child then calls setpgid() in order to detach itself from the process group.
Kernel places child in its own process group.
(8) Child uses `tcsetpgrp` system call in order to become foreground process group
(9) Kernel makes child’s process group into the foreground process group.
(10) Child uses exec system call in order to run the "ls" process
(11) The “ls” process sends write request to the kernel.
(12) Since “ls” process part of foreground process group TTY driver sends the write to the terminal
(13) “ls” process finishes executing and exits
(14) Shell figures out "ls" is finished (how?) and then regains control of terminal by calling tcsetpgrp()
(15) Kernel makes it the foreground process group
(16) Shell reprints prompt and awaits next command

```
bash>
folder
type
bash>
```
Other things to think about

- What is the shell doing while “ls” is running?
- Where does signal handling come into play?
- What do we do if a user wants to start a background process (e.g. “ls&”)?
- When does reaping come into play?
- Is it possible to run a shell within a shell?
Hints

• Start now if you haven’t started yet
  – This is probably the first time you will be dealing with multiple processes, asynchronous signal handling, and system calls

• If you are having trouble understanding the concepts, draw pictures!
  – Visualizing things helps immensely when dealing with complex ideas