Introduction to Linux

January 2011

Don Bahls

User Consultant (Group Leader)

bahls@arsc.edu

(907) 450-8674

Overview

- The shell
- Common Commands
- File System Organization
- Permissions
- Environment Variables
- I/O Redirection and Pipes
- Shell Special Characters

Practice

- **Exercises are available at the following link. You may want to do these exercises after class:**
  - **http://people.arsc.edu/~bahls/classes/exer.tar.gz**

The shell

- **A shell is a program which lets you interact with a system.**
- **It lets you:**
  - run programs.
  - interact with files, directories and devices on the system.
  - “see” what those programs are doing.
  - send signals to other programs running on the system.
  - and more- like setting environment variables, setting limits, etc…

Some common shells

- `sh` **- bourne shell**
- `ksh` **- korn shell**
- `bash` **- bourne-again shell**
- `csh` **- C shell**
- `tcsh` **- tenex C shell (a.k.a turbo C)**

- **Basic functionality is similar.**

What shell should you use?

- **`tcsh` and `bash` probably the easiest to learn for beginners, however `ksh` is the default at ARSC for historical reasons.**
- **With `tcsh` and `bash`:**
  - command **history** (i.e. previously run commands) can be accessed using the **up** and **down** arrow keys.
  - the **tab** key tells the shell to try to perform filename completion.
  - there's a lot more, but this will get you started.

How do you change your shell?

- **On many systems the `chsh` command will let you change shells.**
- **If that doesn't work, talk to your help desk or system manager.**

Common Commands

File Related

- `ls`
- `cd`
- `mkdir`
- `rmdir`
- `rm`
- `pwd`

Access Related

- `chmod`
- `chgrp`
- `groups`

Process Related

- `ps`
- `kill`

General Purpose

- `more/less`
- `grep`

Documentation

- `man`
- `info`

Advanced Topics

- `pushd, popd, alias,…`
- `…time permitting…`

Common Commands - continued

- **We won't cover all of these commands, but by the end of this talk you'll know where to find more information on all of them.**
- **Almost all of the aforementioned commands are separate executables (however `cd` is a built-in shell command in many shells).**
- **NOTE: Most UNIX environments are case sensitive, so "`ps`" is not the same as "`PS`","`Ps`" or "`pS`"!**

man - on-line manuals

- **man pages are available for most system commands in UNIX**

```
LS(1)                          User Commands                          LS(1)

NAME
       ls - list directory contents

SYNOPSIS
       ls [OPTION]... [FILE]...

DESCRIPTION
       List  information  about  the FILEs (the current directory by default).
       Sort entries alphabetically if none of -cftuSUX nor --sort.

       Mandatory arguments to long options are  mandatory  for  short  options
       too.
```

`man` - on-line manuals

- **View the man page for `ls`**

```
man ls
```

- **The "`-k`" flag lets you search for a term in all man pages.**

```
man -k color
```

- **If there isn't a man page, try using the "`-h`" or "`--help`" options with the command:**

```
ls --help
```

`man` - on-line manuals

- **Let's look at a real man page...**
- **A few tips:**
  - press the "**spacebar**" key to move to the next page.
  - press the "**enter**" key to move to the next line.
  - press "**/**" to get into search mode.
    - Then type a string to search for.
    - The "**n**" key will find next occurrences of the string.
  - press "**q**" to quit

`info` - on-line manuals

- **May contain more in-depth documentation. {e.g. info emacs}**
- **A few tips: {Emacs-like navigation}**
  - use "**arrow**" keys to navigate current page.
  - press the "**enter**" key to drill down into topics.
  - press "**[ ]**" keys to move forward/back pages.
  - press "**t u**" key to top of topic or move up the topic.
  - press "**p n**" key to go to the previous or next topic.
  - press "**/**" to search and repeat search.
    - Then type a string to search for.
  - press "**ctrl-C**" to quit

pwd - print working directory

- **prints the directory the shell is in.**

```
% pwd
/home/user
```

`ls` - list directory contents

- **The `ls` command shows information about a file or directory.**
- **Basic Command**

```
% ls
bin
```

- **Long Listing**

```
% ls -l /usr/bin/g++
-rwxr-xr-x  4 root root 109768 May 19  2005 /usr/bin/g++
```

`ls` - list directory contents

- **Show long listing of all files**

```
% ls -la ~
total 72
drwxr-xr-x  3 user  staff 4096 May 11 17:42 .
drwxr-xr-x  5 root  root  4096 May 11 17:41 ..
-rw-r--r--  1 user  staff 24 May 11 17:41 .bash_logout
-rw-r--r--  1 user  staff 191 May 11 17:41 .bash_profile
-rw-r--r--  1 user  staff 124 May 11 17:41 .bashrc
drwxrwxr-x  2 user  staff 4096 May 11 17:42 bin
-rw-------  1 user  staff 67 May 11 17:41 .Xauthority
```

`ls` - list directory contents

- **Notice files beginning with "." are now included in the output now.**
- **Also notice the special directories**
  - "." the current working directory
  - ".." the parent directory
- **Another special directory**
  - "~" refers your home directory

`cd` - change directory

- `cd` **lets you change the working directory of the shell.**
- `cd` **with no parameters returns you to your home directory.**
- **Examples**

```
cd
cd /usr/bin
cd ..
```

- **Can you think of another way to get to your home directory?**

`mkdir` - make directory

- `mkdir` **creates a new directory**

```
% mkdir new_dir
```

- **sometimes commands fail.**

```
% mkdir ../../new_dir
mkdir: cannot create directory `../../new_dir': Permission denied
```

- **you must have permissions to write and remove files and directories**

  (more in the Permissions section)

`rmdir` - remove directory

- `rmdir` **removes a directory.**
- **The directory must be empty to be removed**

```
rmdir new_dir
```

`rm` - remove files or directories

- `rm` **removes a file or directory.**
- **there's no way to undo a** `rm`**!**
- **be careful!**
- **remove file**

```
rm filename
```

- **recursively remove a directory.**

```
rm -r new_dir
```

`more` - paging filter

- `more` **lets you display files to screen one page (i.e. screen full) at a time.**

```
% more ~/.bashrc
# .bashrc

# User specific aliases and functions

# Source global definitions
if [ -f /etc/bashrc ]; then
        . /etc/bashrc
fi
```

`more` - paging filter

- `more` **also lets you view output from another program one page at a time.**

```
ls -l /usr/bin | more
```

- **The "`|`" symbol is called a pipe. A pipe lets you connect the output from one program to the input of another program (more in the I/O section).**
- **You might prefer the `less` command (available on many systems).**

`ps` - process status

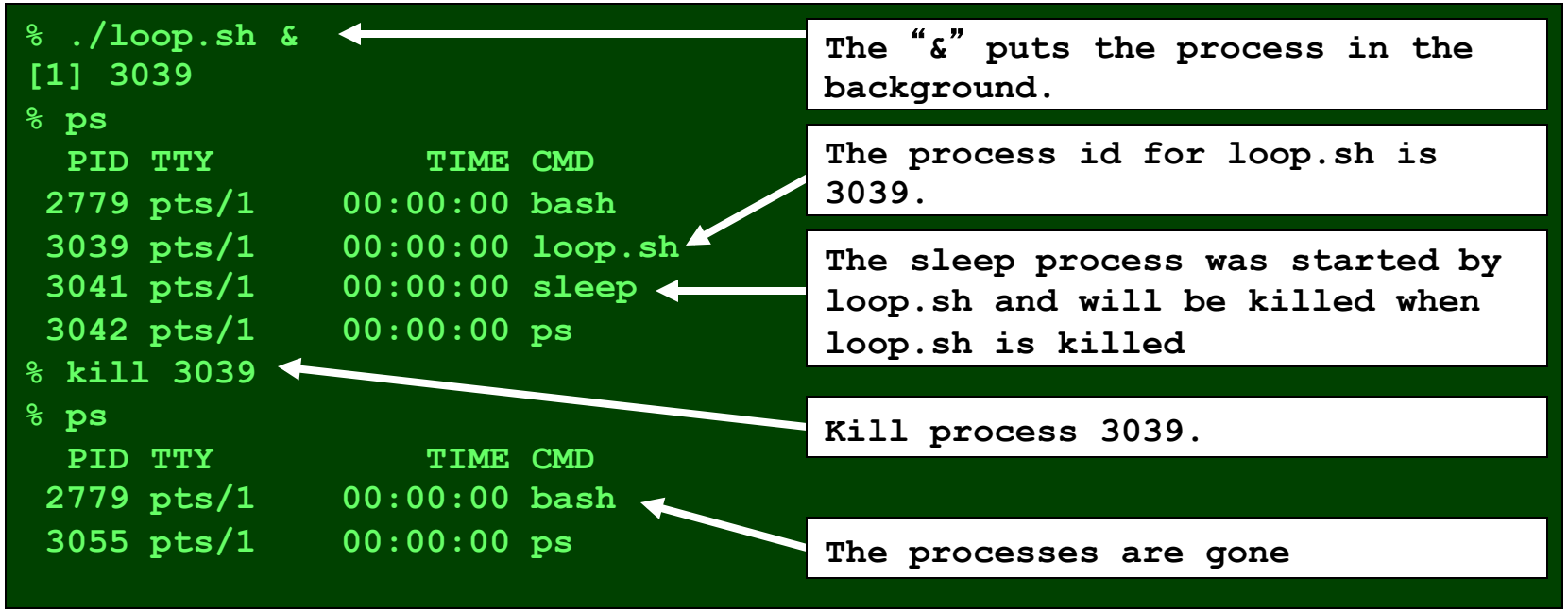
- **Every running process (i.e. program) has an associated process id.**

```
% ps
  PID TTY          TIME CMD
 2779 pts/1    00:00:00 bash
 2810 pts/1    00:00:00 ps
```

- **The default output for `ps` shows only child processes of the shell. Try "`ps -elf`" or "`ps -aux`" to get all processes**

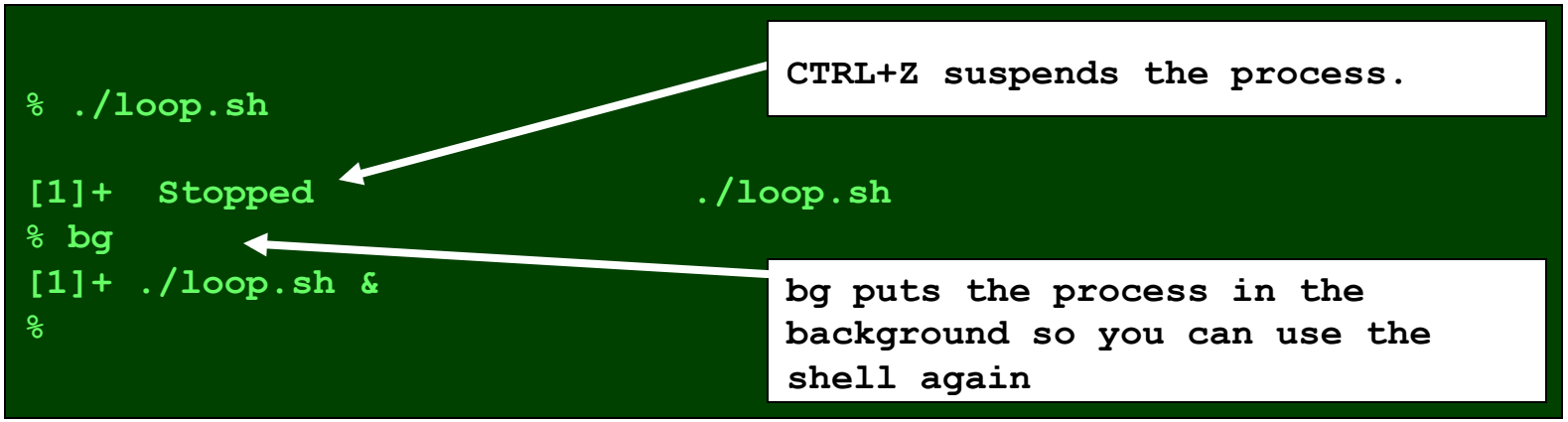
`kill` - terminate a process

- **The `kill` command lets you send a signal to a process id.**



Related Shell Operations

- **Pressing CTRL+C sends a signal to the current running process, just like `kill`.**
- **Pressing CTRL+Z sends a suspend signal. "`bg`" and "`fg`" let you put suspended process in the background and foreground.**



`kill` - continued

- **Kill can send a number of different signals. Sometimes processes might not respond to a particular signal. If all else fails, "`kill -9`" (a.k.a. "`kill -KILL`") should work:**

```
kill -9 3039
```

- **NOTE: you should only use "`kill -9`" if other signals don't work!**

File system Organization

- The directory "/" is called the root directory in UNIX. All other directories are located under this directory.
- Some of these directories have actual files in them, others provide access to hardware devices and other system information.

Common Locations

- `/bin` **- executables**
- `/usr` **- executables, include files, man pages, libraries and more.**
- `/etc` **- system settings files**
- `/home`**,** `/Users`**,** `/u1`**,** `/u2` **- home directories can be in a lot of different locations depends on the OS and the admin who is running the machine. (Your home directory is the directory you enter when you log in).**

Where's my home directory?

```
% cd
% pwd
/home/user
% cd ~
% pwd
/home/user
```

Permissions

- **UNIX uses permissions to control access to files and directories. There are three categories of permissions**
  - user permissions
  - group permissions
  - other permissions

Permissions - continued

- **Each permissions category has three attributes.**
  - read
  - write
  - execute

An Example

```
% ls -l /usr/bin/g++
-rwxr-xr-x  4 root root 109768 May 19  2005 /usr/bin/g++

-                        (this is a regular file)
 rwx           root      (permissions and file owner)
    r-x             root (group permissions and group)
       r-x               (other permissions)
```

- "root" owns this file, and can read, write and execute.
- the "root" group can read and execute this file.
- everyone else can read and execute this file too.

Another Example

```
% ls -l ~
drwxr-x---  4 fred staff 109768 May 19  2005 bin

d                          (this is a directory)
 rwx           fred        (permissions and directory owner)
    r-x              staff (group permissions and group)
       ---                 (other permissions)
```

- "fred" owns this directory, and can read, write and execute.
- the "staff" group can read and execute this directory.
- no one else can access this directory.

Permissions Commands

- **The "`groups`" command shows which groups you are in.**
- **The "`chmod`" command lets you change permissions.**

```
% chmod g+rx ~/bin
```

Add group read and execute permissions to ~/bin (bin directory in ~)

```
% chmod 750 ~/bin
```

Set user read, write, and execute permissions. Also set group read and execute permissions.

4 = Read
2 = Write
1 = Execute

```
                          r   w   x
(u)ser permissions        4 + 2 + 1 = 7
(g)roup permissions       4 + 0 + 1 = 5
(o)ther permissions       0 + 0 + 0 = 0
```

Security

- It's a very bad idea to give world (i.e. other) write permissions. Anyone with access to the system could change the file on you.
- Some dot files/directories contain sensitive information. Be careful who you give access to.

Environment Variables

- **Environment variables store short strings of information that can be read by child processes.**
- **Some important variables:**
  - `PATH`: Where the shell looks for executables. This lets you enter "`ls`" instead of "`/usr/bin/ls`".
  - `HOME`: Set to the path of your home directory.

ARSC Specific Environment Variables

- `SCRATCH`**: Temporary directory with fast local disk.**
- `WORKDIR`**: Temporary directory with fast shared disk (or local disk).**
- `ARCHIVE_HOME`**: Long term storage**
- **Example Use:**
  - `ls $WORKDIR`
  - `mkdir $SCRATCH/mydir`
  - `cd $ARCHIVE_HOME`
- **Your shell will expand environment varibles.**
  - For Example:

    `$ARCHIVE_HOME` expands to `/archive/u1/uaf/username`

Environment Variables - Continued

- **The "`env`" command show all of the environment variable that are set.**
- **You can also show an individual environment variable using the "`echo`" command:**

```
% echo $PATH
/bin:/usr/bin:/usr/local/bin
```

PATH

- **The order of directories in the `PATH` is important. The shell searches for executables in the order they are found in the `PATH` environment variable.**
- **The current directory (i.e. “ . ”) is typically not in the `PATH` for security purposes.**

“ ”
.

- **To run an executable in the current directory, you need to include the “ . ”**
- **To run “myprog” in the current directory:**

```
./myprog
```

or include the full path to the executable:

```
/home/user/myprog
```

- **If you decide to add “.” to your path, put it at the end.**

Setting an Environment Variable

- **Appending a directory to the PATH**
  - **csh / tcsh syntax**

```
setenv PATH ${PATH}:${HOME}/bin
```

  - **ksh / bash syntax**

```
export PATH=$PATH:$HOME/bin
```

- **Setting an environment variable**
  - **csh / tcsh syntax**

```
setenv FRED “hello”
```

  - **ksh / bash syntax**

```
export FRED=“hello”
```

I/O Redirection and Pipes

- **UNIX programs have three forms of standard I/O**
  - stdin: input, normally from the keyboard
  - stdout: output, normally to the screen
  - stderr: error output, normally to the screen
- **However I/O can be redirected.**

Redirecting I/O

- **Redirecting stdout to a new file**

```
ls > ls.out
```

- **Redirecting stdout appending to a file**

```
ls >> ls.out
```

- **Sending a file to stdin**

```
./myprog < input
```

- **Stderr redirection depends on the shell we won't cover it here.**

Using Pipes

- **With pipes, programs using stdin and stdout can be tied together so that the input from one command comes from the output of another.**

```
more myfile | wc -l
cat people | sort -u | wc -l
```

Shell Special Characters

- **Some characters are interpreted in special ways by the shell.**
  - "`*`" matches anything

    ```
    ls /usr/bin/g*
    ```
  - "`?`" matches a single character

    ```
    ls /usr/bin/g??
    ```
  - "`&`" puts a process in the background so you can continue to use the terminal.

    ```
    ls -l /usr/bin > ls.output &
    ```

Shell Special Characters

- **Some other special characters that we've already seen.**
  - "$>$" stdout redirection
  - "$<$" stdin redirection

Fortran and Linux

- **From http://gcc.gnu.org/wiki/:**

  Above all.... Google on "fortran 95 tutorial" and you'll find every style and language under the sun!

- **The same goes for “unix/linux tutorial”**

Parting words

- **Don't forget about the `man` and `info` commands!**
- **Remember to search online**
- **Get The Exercises:**

```
curl http://people.arsc.edu/~bahls/classes/exer.tar.gz >
  exer.tar.gz
tar –zxf exer.tar.gz
cd exer
more Exercises
```

Appendix A - Command Reference

- **Kerberos Commands**
  - `kshell`
  - `kinit`
  - `krlogin`
  - `krcp`
  - `kftp`
- **Openssh**
  - `ssh`
  - `sftp`
  - `scp`
- **Compilers/Interpretors**
  - `gcc / cc`
  - `c99`
  - `g++ / c++ / CC`
  - `gfortran / f95 / pgf90`
  - `python`
  - `perl`
  - `ruby`
  - `ld`

Appendix A

- **Utilities**
  - awk
  - cat
  - cut
  - diff
  - find
  - head
  - gawk
  - grep
  - gunzip
  - gzip
  - ld
  - ldd
  - less
  - egrep
  - sed
  - sort
  - tail
  - tar
  - uniq
  - wc
  - which
- **Text Editors**
  - emacs
  - gedit
  - nano / pico
  - nedit
  - vi
  - vim / gvim

Appendix B - Shell Reference

- `tcsh` **dot files are all located in the home directory of the user.**
  - The `.login` is read on login.
  - The `.tcshrc` (or `.cshrc`) is read when each new shell is spawned.
  - The `.logout` is read on logout.

Appendix B

- `bash` **dot files are all located in the home directory of the user.**
  - The `.bash_login` or `.bash_profile` or `.profile` is read on login.
  - The `.bashrc` is read when each new shell is spawned.
  - The `.bash_logout` is read on logout.

```
% pwd
/Users/cermak
% pushd /usr/local
/usr/local ~
% pushd /tmp
/tmp /usr/local ~
% dirs
/tmp /usr/local ~
% popd
/usr/local ~
% pwd
/usr/local
% popd
~
% pwd
/Users/cermak
```

Appendix C

Directory navigation

pushd – put current directory on the stack and change to new directory

popd – pop last directory off the stack and change to it

dirs – show directory stack