CST8207 – Linux O/S I

The Shell

Topics

- Kernel (briefly)
- Shell
- What happens when you hit [ENTER]?
- Output redirection and pipes
- Noclobber (not a typo)
- Shell prompts
- Aliases
- Filespecs
- History
- Displaying file contents

Associated Readings

- Chapter 1 (parts of)
- Chapter 7 (The entire chapter)
- Chapter 9 (The entire chapter)
- Chapter 5 (utilities, up to end of Page 160)

Kernel

- Kernel is the core part of Linux. It manages resource of Linux. Resources means facilities available in Linux. For e.g. Facility to store data, print data on printer, memory, file management etc.

- Kernel decides who will use this resource, for how long and when. It runs your programs (or set up to execute binary files).

- The kernel acts as an intermediary between the computer hardware and various programs/application/shell.

Kernel

- It performs the following tasks :
  - I/O management
  - Process management
  - Device management
  - File management
  - Memory management

Introduction to Shells

Q: Who's on first?
A: "Who" is on first.

See youtube.com, search for "who's on first" Abbot, Costello
Try to indentify the problem they have

Shell

- In OS there is special program called Shell. Shell accepts your instructions or commands and pass them to kernel.
- Shell is not part of system kernel, but uses the system kernel to execute programs, create files etc.
- In Unix and Linux, “shell” is the term used to describe the command-line interpreter (roughly equivalent to command.com in DOS).

Shell

- Several shell available with Linux including: BASH, CSH, TCSH, KSH
- In this course, we will use the bash shell because of its wide popularity
- BASH (Bourne-Again SHell), developed by Brian Fox and Chet Ramey from Free Software Foundation (FSF). It is the most common shell in Linux.

Shell

- To find all available shells in your system type following command:
  cat /etc/shells

- To find your current shell type following command
  echo $SHELL

- Builtins
  - A builtin is a utility that is built into a shell. Each of the shells has its own set of builtins.
  - A shell does not fork a new process when you execute builtins, so they run more quickly.

Commands, Programs, and Other Stuff

Command
A directive to the shell typed at the prompt. It could be a utility, another program, a built-in, or a shell script.

Program
A file containing a sequence of executable instructions. Note that it's not always a binary file but can be text.

Filter
A command that can take its input from ***stdin*** and send its output to ***stdout***. It normally transforms a stream of data through a series of pipes. All good scripts should be written as filters.

Utility
A (set of) program(s) that provides services to a user.

Commands, Programs, and Other Stuff (cont.)

Built-in
A command that is built into the shell. That is, it is not a program as defined above. They do not require a new process unlike those above.

History
A list of previous commands that can be recalled, edited if desired, and re-executed.

Token
The smallest unit of parsing; often a word delimited by white space (blanks or spaces, tabs and newlines) and other punctuation (special characters).

Commands, Programs, and Other Stuff (still cont.)

Stdin
The standard input; the keyboard; what **cin** reads from; the file at offset 0 in the file table.

Stdout
The standard output; the screen, what **cout** writes to; offset 1 in the file table;

Stderr
The standard error; the screen; what **cerr** writes to; offset 2 in the file table.

Standard I/O
stdin, **stdout**, and **stderr**.

Commands, Programs, and Other Stuff (still cont.)

Pipe
To use a shell service that connects the **stdout** of one program to the **stdin** of the next; the "|" symbol.

Redirect
To use a shell service that replaces **stdin**, **stdout**, or **stderr** with a regular file.

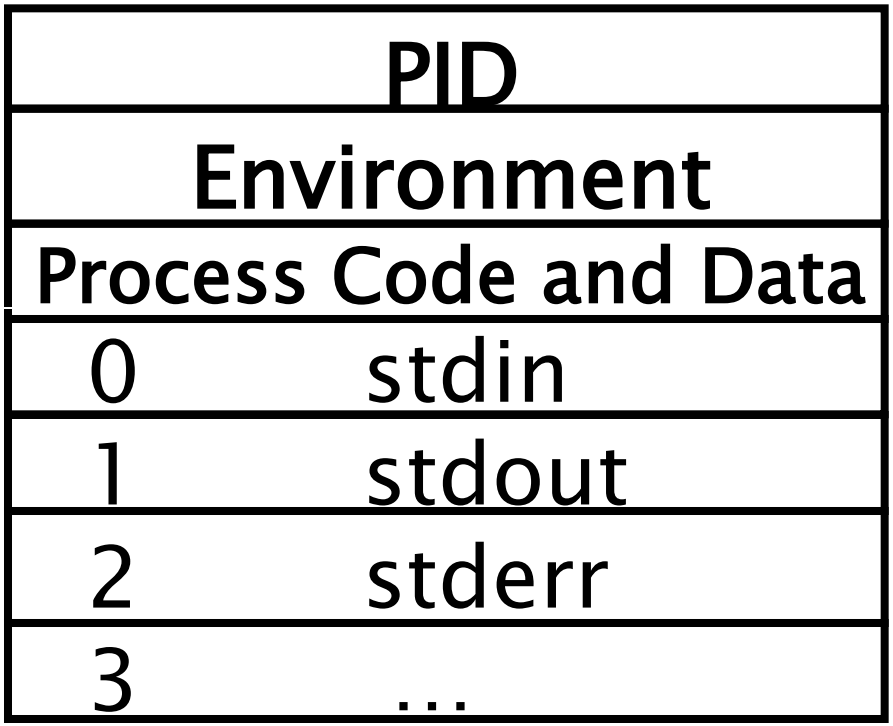
Processes

- A process is a program that is being executed.
- Some processes (called **daemons**) never end, as they are providing a service to many users
  - such as FTP services from **ftpd**.
- Other processes are run by you, the user, from commands you enter at the prompt.
- These usually run in the foreground, using the screen and keyboard for their **standard I/O**. You can run them in the background instead, if you wish.
- Each process has a PID (or pid, a process identifier), and a parent process with its own **pid**, known to the child process as a **ppid** (parent pid). You can examine the running processes with the **ps** command or examine the family relationships with **pstree**.

Child Process

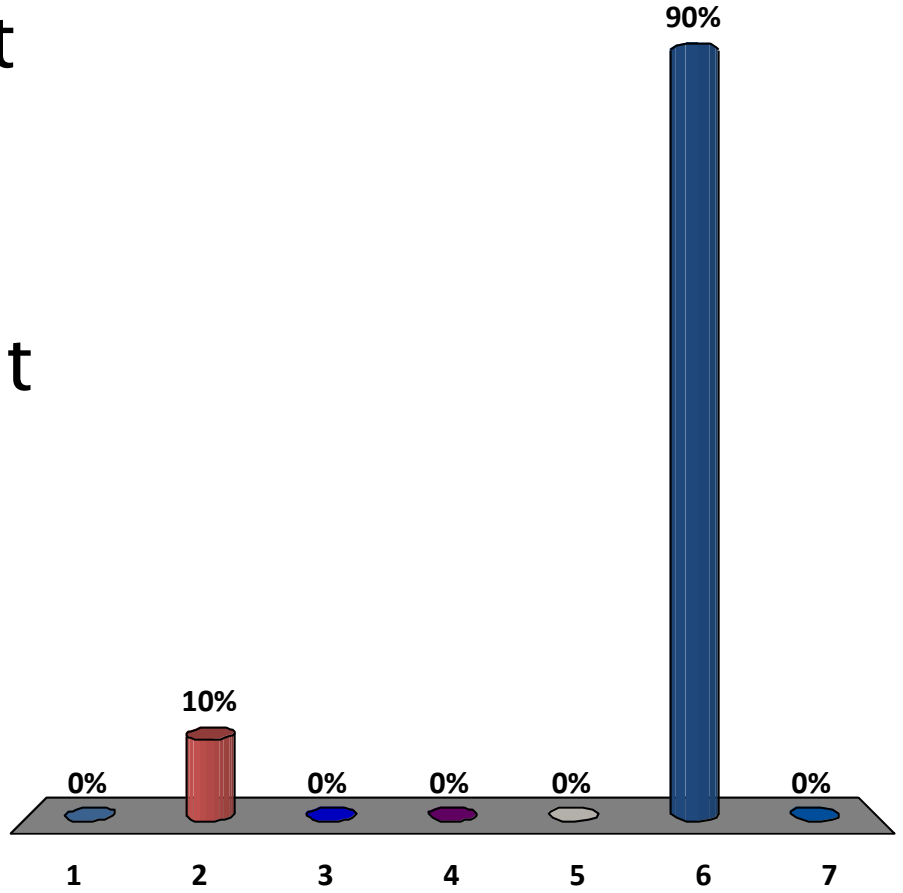
- A child process is **forked** or **spawned** from the parent by means of a system call to the kernel services.
- Forking produces an exact copy of the process, so it is usually then replaced during an exec system call (ie the exact copy, the child, usually asks the kernel to turn it into a different process).
- The forked copy also includes the environment variables and the file table.
- This latter becomes very useful when redirecting the standard I/O, since a child can redirect its I/O without affecting the parent.

Representation of a Process



What does the Kernel do?

1. I/O management
2. Process management
3. Device management
4. File management
5. Memory management
6. All of the above
7. None of the above



Parsing the command line

There's the system prompt. You enter a line of words, giving a command to the shell. What happens next?

Parsing the command line

1. Substitute **history**
2. Tokenize command line (break it into "words" based on spaces and other delimiters; also known as lexical analysis, or parsing)
3. Update **history**
4. Process quotes
5. Substitute aliases and define functions
6. Set up redirection, pipes, and background processes
7. Substitute variables
8. Substitute commands
9. Substitute filename (called "globbing")
10. Execute (run) program

History

The command history is a list of the previous commands you have executed in this session with this copy of the shell. It's usually set to some large number of entries, often 100 or more.

History

- To list the history:

  `System prompt> history | more`
- To repeat the last command entered:

  `System prompt> !!`
- To repeat the last `ls` command:

  `System prompt> !ls`
- To repeat the command from prompt number 3:

  `System prompt> !3`
- To scroll up and down the list (arrow keys):

  `System prompt>` ↑↓
- To edit the command:

  → [DEL] [Backspace]

Redirection

- Three files are available immediately upon shell startup: stdin, stdout, stderr
- These can be overridden by various redirection operators
- Following is a list of these operators

Redirection Operator	Behaviour
< filename	Redirects input from filename
> filename	Redirects output to filename
>> filename	Appends output to filename
2> filename	Redirects errors to filename
2>>filename	Appends errors to filename
&> filename	Redirects output and errors to filename
>&	As above; preferred format
2>&1	Redirects errors to output destination
1>&2	As above but reversed
>!	Overrides noclobber
<> filename	Uses filename as both standard input and output if a device file (from /dev)

Command Aliases

- To create an alias

  ```
  alias ll="ls -l"
  ```

- To list all aliases

  ```
  alias      or  alias | more
  ```

- To delete an alias

  ```
  unalias ll
  ```

Substitute Filenames: Metacharacters

Metacharacter	Meaning
\	Interprets the following character literally
&	Processes in the background
;	Separates commands
$	Substitutes variables
?	Matches for a single character
[abc]	Matches for one character from a set; in this case an a or b or c
[!abc]	Matches for one character **not** from a set; in this case not an a or b or c
*	Matches zero or more characters
(cmd)	Executes commands in a subshell
{cmd}	Executes commands in the current shell

Wildcard Characters

*	Matches 0 or more characters in a filename
?	Matches 1 character in a filename
[aed]	Matches 1 character in a filename- provided it is either a,e, or d.
[a-e]	Matches 1 character in a filename- provided it is a,b,c,d, or e
[!a-e]	Matches 1 character in a filename- provided it is not a,b,c,d, or e

Output Redirection

- Output Redirection Operator (>)

  ls > file1 (Redirect output to a file)

  cat /etc/fstab > file2 (Redirect output to printer)

- Append Redirection Operator (>>)

  Lets you redirect a output to the end of an existing file

  ls >> file3 (Add output to the end of a file)

Pipe

- Using “|” to connect two commands
  - *cat /etc/passwd | grep “user1”*
  - The output of the first command is the input of the second command

tee command

- It sends output to both standard output and a file:

- Examples:

  cat /etc/fstab | tee ~/file1

  ls -a ~/ | tee -a ~/file1

  - **Option -a : append to given file, not to overwrite**

Not to Overwrite Files

- Setting **noclobber** variable prevents you from accidentally overwriting a file when you redirect output to a file
  - To setup noclobber

    set –o noclobber
  - To unset noclobber

    set +o noclobber
- Example:

  date > f1

  set –o noclobber

  date > f1

  set +o noclobber

  date > f1

Command Line Editing

- Ctrl+H
  - To erase one character
- Ctrl+U
  - To erase one line
- Ctrl+W
  - To erase a word

Customizing your prompt

- **Changing primary user prompt:**
  - PS1 environment variable
    - You can make your own prompt by using the following codes.

      \d, \h, \u, \n, \s, \t, \w

      PS1='[\u@\h \w] $'
- **Changing secondary user prompt**
  - PS2 environment variable
  - Default secondary prompt is ">" for unfinished command in previous line, waiting for user to continue the command line.

    PS2="secondary prompt: "

wc command

- ***wc*** command prints nline, word and byte counts for files
  - Options
    - w: word counts
    - l: line counts
    - c: character counts
  - Examples:
    - **wc ~/***
    - **wc –l /etc/fstab**

Using Brace Expansion:{}

- Using {} to create a list of files or directories
  - touch ~/dir1/{f1,f2,f3}
  - mkdir ~/dir2/{a1,a2,a3}
  - mkdir -p \
    ~/backup/{old,new}/{labs{1,2,3},lecture{1,2,3}}
    - How many directories have been created?
      - find ~/backup -type d | wc -l

History Command

- *history*
  - Keeps a record of the most recent commands you have used
  - Clear history list
    - *history -c*
  - Configuring history
    - HISTFILE: where the history is stored, by default, this file is ~/*.bash_history*
    - HISTSIZE: the number of history events could be saved

History Event References

- You can use *!* Command to reference an history event
  - The reference can be the event number
    - *!3*
  - The reference can be a beginning set of characters in the event
    - *!chm*
  - The reference can be an offset from the end of the history list
    - *!-4*

Displaying the Contents of Text Files

tail	Display the last 10 lines
head	Display the first 10 lines
more	Display text files page by page/line by line
less	Less is more
cat	Display the entire file
tac	Display the entire file in reverse order

df and *du* commands (Reminder)

- *df* command
  - Used to check free space on the disk
  - Only view mounted filesystems
  - Options: -h
  - Example: *df -h*

- *du* command
  - Directory usage: to view the size of directory and its contents in kilobytes.
  - Options: -s, -h
  - Example: *du -hs /etc*