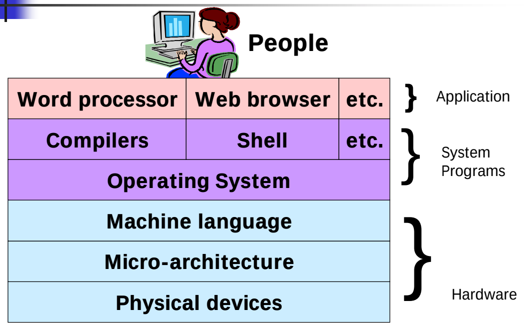
Unix Basics

Stanley Yao
Computer Science Department
University of Arizona

Outline

- <u>OS Basics</u>
- Shell Basics
- Process
- I/O, Redirection & Pipe

What is Operating System?



OS Concepts

- OS: protected library of useful functions
- OS Functions:
  - Standard library: provide standard facilities (abstractions) that everyone needs.
  - Coordinator: allow several things to work together in efficient and fair ways (resource management).

UNIX

- UNIX is a multiprogrammed, timeshared operating system
  - Multi-programmed: OS runs multiple processes simultaneously
  - Process: running program
  - Timeshared: multiple users share the system simultaneously

OS Families

- Unix
  - AT&T: System V
  - Berkeley: BSD
  - Sun: Solaris, SunOS
  - IBM: AIX
  - etc.
- Microsoft Windows
  - Win2000, WinNT, WinXP
- MacOS
- And many many more …

Unix & C Programming Language

- Unix is written in C
- C is a programming language on Unix
- Who comes first?
- Programming languages .vs. Human Languages

Outline

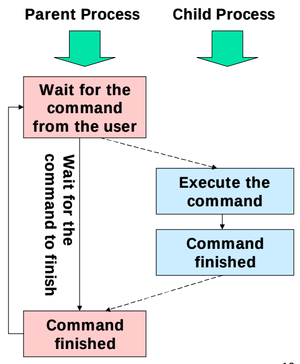
- OS Basics
- Shell Basics
- Process
- I/O, Redirection & Pipe

Shell

- What is "Shell"?
  - Command-line interface to the OS
    - Interpreter
    - Environment
  - A separate program, and not part of the OS
- What shells are there in UNIX?
  - C Shell
  - Bourne Shell
  - bash
  - tcsh
  - etc.

C-Shell

- Invented by Bill Joy
- Commands
  - Built-in commands
  - Programs
    - Executables
    - Shell scripts



Command format

- command [command args] [shell args]
  - Command can be a built-in or a file name
  - For a file name, where to find the file? PATH
- Example:
  - **`ls /home/yao`**
  - **`cat schedule > mysched`**

Command Helpers

- "**`which`**" command will tell you where the command is found in the path.
- "**`man`**" command can get the information about commands and standard C library functions.
- "**`command -h`**" or "**`command –help`**" to get brief summary of the usage of "command" in GNU release

Multiple Commands

- cmd_1; cmd_2; …; cmd_n
  - Execute sequentially
  - Prompt returns when the last one finishes
  - No waiting among commands
  - Appropriate for a set of commands that forms a logical group
- Example:
  - **`mkdir proposal; cd proposal; ls`**
  - **`cd utils; cc –o gets gets.c; cd ..`**

Command Groups

- (cmd_1; cmd_2; …; cmd_n)
- Executing without affecting the current environment
  - Environment: the set of characteristics describing a user's working area
- Creating a subshell
- Example:
  - **`(cd utils; cc –o gets gets.c)`**

Environment Variables

- User environment: special information, e.g. login directory, your mailbox, your terminal type, etc.
- Environment variables: maintains the special environment information mentioned above
- Can be passed to programs executed from the shell and affect the programs' behavior, e.g. default printer, mailbox, TERM in vi

Environment Variables (cont.)

- **`setenv VARNAME string`**
- **`unsetenv VARNAME`**
- Use: **`$VARNAME`**
- Show all variables: **`env`** (or printenv)
- Conventionally all cap
- Examples:
  - **`PATH`**
  - **`HOME`**

Shell Variables

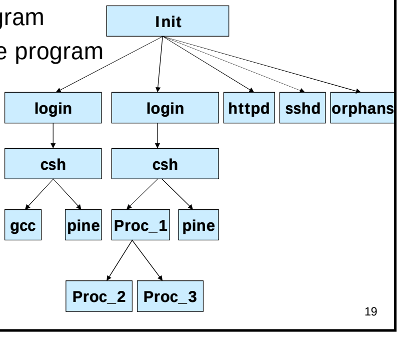
- Internal to the shell
- **`set name = value`**
- **`unset name`**
- Use: **`$name`**
- Show all variables: **`set`**
- Examples:
  - path: kept sync by shell
  - noclobber: prevents redirection from overwriting an existing file. (wc foo.c >! /tmp/out)
  - status

Outline

- OS Basics
- Shell Basics
- Process
- I/O, Redirection & Pipe

Unix Process Basics

- Process
  - Running program
  - Run the same program twice?
- Process tree
  - ps: ps aux
  - pstree



Standard I/O for Each Process

- Standard I/O connections (*file descriptors*)
  - *stdin*: standard input (e.g. terminal keyboard)
  - *stdout*: standard output (e.g. terminal screen)
  - *stderr*: standard error (e.g. terminal screen)
- A typical UNIX process reads its input from stdin, writes its output to stdout, and any error messages to stderr.

Standard I/O for Each Process (cont.)

- Inherited from the parent
- Shell can control the file descriptors of its children (redirection)
- This is a very powerful feature of UNIX, as it means the shell can make a process read/write different things without modifying the program, e.g. files, devices, another process.

Process Return Value

- A UNIX process can return a status value to its parent when it exits
  - Zero: OK
  - Non-zero: Error
- The parent uses the value to decide what to do next.
- The C-shell stores this value in a variable called **`status`**.

Outline

- OS Basics
- Shell Basics
- Process
- I/O, Redirection & Pipe

Why redirection or pipe?

- You want to store the result in a file instead of print them out on the screen
- You want to prepare the input in a file instead of typing them every time the program is run
- You want to connect several UNIX tools in a chain to finish a more complex work. The data flow through those tools. The output of a previous tool will be the input of the next.

Redirection

- Default standard I/O: the same as the shell
- Shell arguments in the command line can specify the change of the standard I/O
- Examples:
  - Write stdout to a file: **`wc foo.c > /tmp/foo`**
  - Append stdout to a file: **`wc foo.c >> /tmp/foo`**
  - Write stdout & stderr to a file: **`wc foo.c >& /tmp/foo`**
  - Ignore stdout: **`wc foo.c > /dev/null`**
  - Read stdin from a file: **`wc < foo.c`**
  - Read stdin from a device: **`wc < /dev/null`**

Output Redirection Summary

	Create/truncate and write	Append
Stdout	>	>>
Stdout+stderr	>&	>>&

Pipe

- cmd_1 | cmd_2 | … | cmd_3
- Connect a program's standard output with another program's standard input
- Example:
  - **`who > temp; wc –l < temp; rm temp`**
  - **`who | wc –l`**
  - **`cat dict1 dict2 dict3 | sort +1 | pr | lpr`**
  - **`cc pcman.c |& more`**
  - **`cc pcman.c |& pr | lpr`**

Pipe (cont.)

- What commands can be connected with pipes?
  - Command on the left must write to the standard output
  - Command on the write must read from the standard input
- Example:
  - **`ls | rm`**
  - Result: Broken pipe line

Acknowledgement

- John H. Hartman, *Classnotes for Csc352-Spring03*, CS Dept., University of Arizona, 2003
- Gail Anderson, Paul Anderson, *The Unix C Shell Field Guide*, Prentice Hall, 1986
- Andrew S. Tanenbaum, *Modern Operating Systems (2nd Ed.)*, Prentice Hall, 2001