The *Linux* Operating System

Charles Abzug, Ph.D.
Department of Computer Science
James Madison University
Harrisonburg, VA 22807

Voice Phone: ***540-568-8746;*** Cell Phone: ***443-956-9424***
E-mail: ***abzugcx@JMU.edu*** OR ***CharlesAbzug@ACM.org***
Home Page: ***https://users.cs.jmu.edu/abzugcx***

Linux

1. Brief History of *Linux*

   a) *Multics* →
   Assembly-Language *Unics* (DEC PDP-7) →
   B-language *Unix* →
   C-language *Unix* →
   →
   →
   AT&T *SVR4*
   →
   BSD *Unix*

2. Open Software Federation

3. *GNU*: "*GNU* is Not *Unix*."

4. The true *Linux* component consists principally of the kernel; remainder is *GNU* utilities and system software.

5. Portability: multiple platforms (Wintel, Sun SPARC, DEC Alpha, Motorola/IBM/Apple PowerPC, others)

Value of *Linux*

1. RELATIVE Smallness
2. RELATIVE Simplicity
3. Robustness and Stability: rareness of system crashes
4. Availability of Source Code: no hassle
5. Modifiability
6. Speed, Responsiveness
7. Security
8. Immunity to PC viruses
9. Dedicated volunteer support team

Deficiencies of *Linux*

1. No centrality of concept
2. No centrality of implementation
3. Hacker culture
4. Arcaneness

Command Line Interpreters ("Shells")

1. Original *Unix* command line interpreter: the *Bourne* shell
   (Stephen Bourne, 1979: AT&T)
   a) Based on Algol
   b) Intended to automate system admin tasks
   c) Concise
   d) Compact
   e) Fast
   f) Default prompt: $

Command Line Interpreters ("Shells") (continued)

2. The *C* shell (primary developer: Bill Joy, 1986: University of California at Berkeley)
   a) Based on the language *C*
   b) Contained enhancements for interactive use:
      i. command-line history
      ii. aliases
      iii. built-in arithmetic
      iv. filename completion
      v. job control
   c) More Complex
   d) More Capable
   e) Slower
   f) Default Prompt: %

Command Line Interpreters ("Shells") (continued)

3. The *Korn* shell (David Korn, 1986: AT&T)
   a) Contained enhancements for interactive use:
      i. editable command-line history
      ii. aliases
      iii. built-in arithmetic
      iv. functions
      v. regular-expression wildcards
      vi. coprocessing
      vii. spepcial debugging features
      viii. job control
   b) Most Bourne shell scripts will run in Korn shell
   c) Not so complex, not so slow as the *C* shell
   d) Runs not only on *Unix*, but also on *OS/2*, *VMS*, and *DOS*.
   e) Default Prompt: %

4. *Public Domain Korn Shell*: free, portable, POSIX-compliant; *pdksh*

Command Line Interpreters ("Shells") (continued)

5. The *TC* shell: an expanded version of the *C* shell, with additional features: *tcsh*

6. The *Z* shell: a *Korn* shell clone, enhanced with many *TC* features (Paul Falsted)

7. The *Bourne Again Shell*: Brian Fox, "Free Software Foundation"): *bash*
   a) Runs *Bourne* shell scripts unmodified.
   b) Adds the most useful features of the *C* shell.
   c) Adds the most useful features of the *Korn* shell

To find out what shells are available on the machine to which you are logged in:

```
$ cat /etc/shells
```

To select a shell (example): `$ chsh`

Uses of a Command Line Interpreter ("Shell")

1. Interprets commands entered interactively at the command prompt.
   a) Searches for the file implementing the command.
   b) Spawns a subprocess to execute it.

2. Interprets and runs Command Procedures (called "Shell Scripts" in UnixSpeak)
   a) Runs through the Command Procedure line by line.
   b) Executes the command of each line in turn as if it had been entered on the keyboard at the command-line prompt (shell prompt).

3. Customizes the user's environment, e.g., by setting variables that:
   a) define the search path.
   b) set default permissions.
   c) set the command-line prompt.
   d) set the terminal type.
   e) set the values of variables required for specific applications:
      i. windows.
      ii. text-processing programs.
      iii. filename and command completion.
      iv. spell checking.
   f) define aliases.

The file /etc/passwd

The file contains environmental variables for each user:

HOME	home directory
SHELL	login shell
USER	login name
LOGNAME	login name
PATH	an ordered list separated by colons indicating the order in which various directories are to be searched for the presence of a file which the user may select later to run

last entry in a line of the file: the name of the program to run at the conclusion of the login procedure (normally this would be a shell program)

Executing a Command-Line Interpreter ("Shell")

To find out what Command-Line Interpreters ("shells") are available on the system:

$ cat /etc/shells

Results on "helium":

```
/usr/bin/sh
/usr/bin/csh
/usr/bin/ksh
/usr/bin/jsh
/bin/bash
/bin/sh
/bin/ash
/bin/bsh
/bin/csh
/bin/tcsh
/bin/ksh
/bin/jsh
/sbin/sh
/sbin/jsh
/aux/bin/bash
/bin/rbash
```

Types of Commands for *Bourne*, *Bash*, and *Korn* Shells

NOTE: These are executed *in the order of preference* specified below:

1. **Aliases: nicknames defined either by the user or system-wide for specific commands.**

2. **Keywords: meanings defined within the command-line interpreter (shell).**

3. **Functions: groups of commands organized as separate routines and stored within the memory of the command-line interpreter (shell).**

4. **Built-In Commands: commands whose execution is defined within the command-line interpreter (shell).**

5. **Executable Programs: resident on disk and located in one of the directories specified in the user's PATH.**

THUS, for example, if the user types the command $ foo the command-line interpreter will first check to see whether "foo" has been defined as an alias. If not, then it will search its list of keywords. If not present here, then it will search its list of functions. If still not present, it will search for it as an internal command within the command-line interpreter. Finally, it will search in order the directories on the PATH for an executable file of that name.

Some Useful Commands

1.	ps au	displays a list of all process running on the system by all users.
2.	pstree	displays all processes running in the form of a tree whose root is the process *init*, the first process that runs on boot-up.
3.	chsh	changes the user's command-line interpreter to be run immediately following login.
4.	ls	provides a listing of the contents of the current working directory.
5.	ls –l	provides a much more informative ("long" format) directory listing.
6.	pwd	prints the name of the current working directory.
7.	cp file1 file2	copies the contents of the file *file1* to a file *file2*.
8.	chmod	changes permissions on files and directories.
9.	umask	changes the default permissions of all subsequently created files and directories.
10.	passwd	changes the user's password.

Entering, Compiling, and Running a Program

1. Use a text editor, such as *vi*, to create your source code file. Name your source file either program1.c or program2.c , as appropriate to the assignment.

2. Compile using the *GNU C* compiler, as follows:

```
gcc –o program1 program1.c

gcc –o program2 program2.c
```

3. Run your program:

```
program1

program2
program2 –s
program2 -1 10 50
```

Instruction Manual for the Use of the *vi* Editor

Please see the following URL: http://www.eng.hawaii.edu/Tutor/vi.html

END