

Objectives

The Linux Operating System

An Overview

Nick Urbanik <nicku(at)nicku.org>
Copyright Conditions: GNU FDL (see <http://www.gnu.org/licenses/fdl.html>)
A computing department

Having completed this module, you will have an overview of a Linux system, including its:

- Underlying philosophy
- System layering — kernel vs. applications
- Core services
- Multiuser and timesharing facilities
- File System
- Network Services
- Desktop and X windowing system

OSSI — Overview of Linux — ver. 1.0 — p. 127

OSSI — Overview of Linux — ver. 1.0 — p. 227

Generic Features of Unix

- Component-based systems
- Very popular with technically skilled
- Not 'solution' oriented
- Building blocks not the building
- Highly network-aware
- Robust, powerful, reliable

OSSI — Overview of Linux — ver. 1.0 — p. 327

Linux — The Kernel of a System

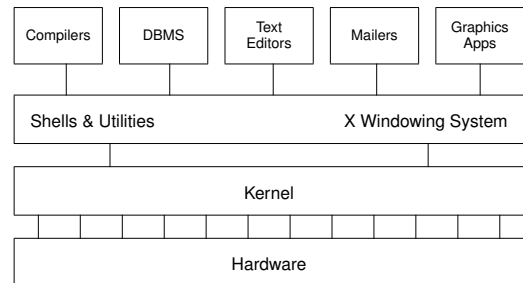


Figure 1: kernel-layering

- What is called Linux is actually a collection of

OSSI — Overview of Linux — ver. 1.0 — p. 427

Fundamental Characteristics of Linux

- Multi-tasking
- Multi-user access
- Multi-processor
- Architecture independence
- POSIX 1003.1 plus basic System V and BSD
- Protected memory mode
- Multiple filesystem types
- Comprehensive networking (TCP/IP and others)
- Multiple executable formats (MS-DOS, iBCS UNIX, SCO, etc)

OSSI — Overview of Linux — ver. 1.0 — p. 527

Multiuser Multitasking and Time-sharing

- Designed as a multi-user system
 - Each user's shells, apps and commands are separate processes
 - Number of simultaneous users limited only by:
 - CPU speed and available memory
 - Min. response times required by users/apps
- Multi-tasking:
 - Many jobs can be under way at the same time
 - Jobs truly *simultaneous* on multi-cpu
- Time-sharing: A single cpu is shared by all processes
 - Processes exec briefly, passing cpu to others
 - *Process switches* occur in milliseconds or less
 - Kernel gives process a sense of total control

Protected memory mode

- Uses the processor's protection mechanisms
- Prevent access to memory already allocated to kernel or other processes
- Bad programs can't crash the system
 - Theoretically

Multiple Filesystem Types

- Native FS is ext3 (Third Extended File System)
 - File names up to 255 chars
 - More secure than conventional UNIX
- Others include:
 - MS-DOS (FAT16), VFAT, FAT32
 - ISO9660 (CD-ROM)
 - HPFS (OS/2)
 - NTFS (Windows NT)
 - reiserfs, XFS, other journaled file systems for Linux,
 - UPS, SysV and other proprietary UNIX
 - NFS (Unix network file system)
 - SMB / CIFS (MS Windows file sharing)

The Many Faces of a GNU/Linux System

- The user may see up to five aspects of Linux:
 - the *filesystem*
 - *processes*
 - the *shell*
 - the X *windowing system*
 - *Inter-Process Communication (IPC)*
- The system is very highly configurable
- Different users may experience totally different views of the same system
- Multiple simultaneous users are normal
 - Linux is designed from the ground up as a *multi-user system*, NOT a 'personal' system

OSSI — Overview of Linux — ver. 1.0 — p. 9/27

Filename

- Maximum length depends on filesystem type
 - Most allow up to 255 characters
- Can use almost any character in a filename, but avoid ambiguity by sticking to:
 - (A-Z) Uppercase letters
 - (a-z) Lowercase letters
 - (0-9) Numbers
 - (.) Full-stop
 - (,) Comma
 - (_) Underscore
 - (-) Hyphen
- Should convey meaningful info about contents
- Type longer filenames using completion for:
 - Filenames

OSSI — Overview of Linux — ver. 1.0 — p. 11/27

Hidden Filenames

- Filenames beginning with a full-stop are *hidden*
- Typically used:
 - To hide personal configuration files
 - To avoid cluttering dirs with rarely used files
- Every dir contains 2 special hidden files:
 - . The current directory file
 - .. The parent directory file

OSSI — Overview of Linux — ver. 1.0 — p. 13/27

Key Features of the Bash Shell

- Command history
- Command aliasing
- Shell scripting
- Filename completion
- Command completion
- Command line editing (*emacs* and *vi* styles)
- Job control
- Key Bindings
- Directory stacking
- Tilde directory notation
- Help function, e.g.

The Filesystem

- The filesystem contains all data in the system
- A name in the filesystem can refer to:
 - a *data file*, which can be:
 - a *plain file*
 - a *directory*
 - a *device* (disk, tape etc.)
 - internal memory
 - OS information (the *proc* system)
- Directories are groups of files
 - Grouped in hierarchical *trees*
- Files are fully specified with their *pathname*
- An original Unix structure; copied by most OSs

OSSI — Overview of Linux — ver. 1.0 — p. 10/27

Filename Extensions and File Types

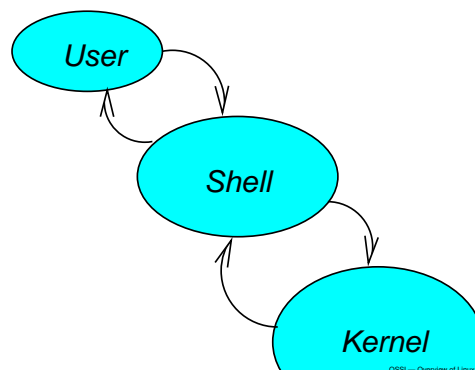
- Filenames *don't* determine other attributes of file, i.e. do not, *automatically*, cause command interpreters to treat them in a particular way
- However:
 - Extensions can enable meaningful naming and automatic file manipulation
 - C compilers and some other programs *do* depend on specific file extensions to carry out particular tasks
- Common conventions for extensions:

Filename	Meaning of Extension
program.c	C programming source file

OSSI — Overview of Linux — ver. 1.0 — p. 14/27

The Shell (bash)

- A *shell* is a program that you interact with



OSSI — Overview of Linux — ver. 1.0 — p. 14/27

Interacting with a Linux 'Terminal'

- Linux can support any number of 'terminal' types
 - nowadays, monitor/keyboard combinations
 - previously, dumb terminals
 - occasionally, printers (debugging servers)
- Most will use the *console* or a windowed terminal, but if not:
 - Linux usually keeps a database of terminal capabilities in `/etc/termcap`^a
 - If your terminal type is not recorded in `/etc/termcap`, you'll have problems running certain programs e.g.
 - cursor driven apps (`top`, `linuxconf`, `vi` etc)
 - The *environmental variable* `TERM` tells programs what terminal type you are using

- True UNIX-like systems treat programs as *tools*
 - Each tool should:
 - Do just one thing well
 - Be generic (untied to specific applications)
 - For new jobs, build new tools
 - (Re-)combine, don't complicate old tools
- Linux can do this because it has:
 - two simple *objects*:
 - the file
 - the process
 - simple methods of *connecting*:
 - processes to files
 - processes to processes

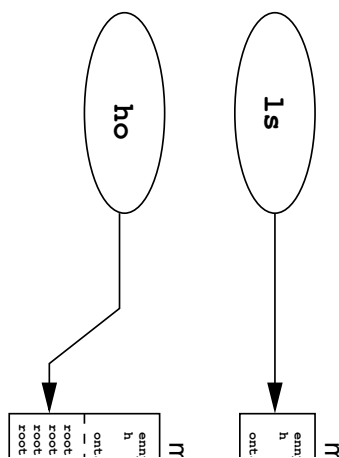


OSSI — Overview of Linux — ver. 1.0 — p. 17/27

Process Communication

- Processes may need to co-operate by
 - sharing files
 - signalling events
 - direct transfer of data
 - pipelines (data streams)
 - synchronising with each other
- Linux provides facilities for:
 - signals
 - shared memory
 - pipes, both named and unnamed
 - semaphores
 - and others
- Processes may use network connections for communication, permitting *client-server* model

OSSI — Overview of Linux — ver. 1.0 — p. 19/27



Pipes & Tools

- Linux tools act as filters:
 - taking data from input streams, modifying it, sending it elsewhere
 - expecting data to come from other tools
 - producing output which *any* other tool can process, e.g. ASCII text
- One tool's output is connected to another's input:
 - *Indirectly*, via a file created by the first tool
 - *Directly*, via a *pipe* or *pipeline*
- For example, to page through a reverse-sorted version of your password file on screen:


```
$ sort -r < /etc/passwd | less
```

- A *program* is an *executable* object, stored in a file
- A *process* is an *executing* object, i.e. ^a
 - an *instance* of a program currently being run
- Existing processes can '*fork*' to create other processes
 - the only way to make new processes
- A user may run multiple copies of same program
- Multiple users may run single/multiple copies
- System tracks *ownership* and *permission*

^aProcesses are often called *tasks*, as in 'multi-tasking'

OSSI — Overview of Linux — ver. 1.0 — p. 18/27

Re-directing I/O to and from Files

- Most processes will take input from the keyboard and output to the screen
- Both input and output streams can be *re-directed* to/from files
- Output to a file (creating or overwriting):


```
$ ls > my-system.txt
```
- Appending output to a file:

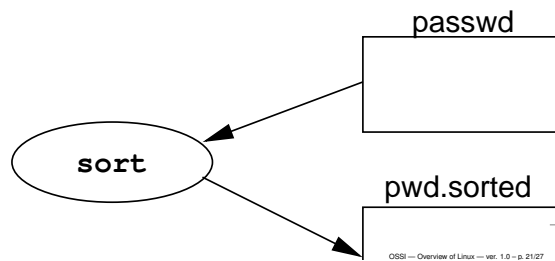

```
$ who >> my-system.txt
```

OSSI — Overview of Linux — ver. 1.0 — p. 21/27

Re-directing I/O to and from Files (continued)

- Take input from one file, output to another:


```
$ sort < /etc/passwd > pwd.sorted
```



OSSI — Overview of Linux — ver. 1.0 — p. 21/27

Linux as a Programming Environment

- *Hierarchical Filestore*
- Extensive set of *powerful tools*
 - for software production, admin and support
- A *common system interface*
 - only one set of procedures to learn
- Processes interface with *anonymous files*
 - programs output to files or devices identically
- *Modular architecture* provides for a completely customised OS, e.g.
 - An OS dedicated solely to graphics rendering
 - A general-purpose system on one floppy
- *Flexible user interface* allows for uniquely customised programming environments

Networking

- Linux is a network operating system.
- The Internet network protocols (TCP/IP) are implemented in the kernel
- Although other media are supported (e.g. radio, infra-red), links are usually across:
 - Ethernet
 - Serial Line (Point-to-point)
- Proprietary file/print serving protocols supported:
 - Appletalk
 - DECNET
 - IPX / Novell Netware
 - SMB / CIFS (MS Windows/NT)

TCP/IP

- A suite of Internet-standard protocols and apps for managing data transfers
- Depicted as a 'stack'
 - hardware and transport control protocols at the bottom
 - user applications (e.g. browsers) at the top
- Client-server apps provide facilities for:
 - Remote login
 - File transfer
 - Resource sharing (e.g. expensive peripherals)
 - Remote command execution
 - Email (internet/intranet/extranet)
 - Web browsing

Documentation

- Copious, but fragmented and/or duplicated

<i>Programmer's Manual</i> /usr/man	The classic ' <i>man pages</i> ', first stop for skilled users, worth learning
info pages	hypertext browsable texts, often identical or updated versions of <i>n pages</i>
/usr/share/doc/ <i>program-name</i>	ascii/html docs installed with the named program
<i>Howtos</i>	Tutorials on Linux-related topics, available on-line if installed (usually in /usr/share/doc)
www	Recently-released programs are usually documented on authorised web sites, many (including older tools) are documented by third-party sites

Table 2: Sources of Linux Documentation

Using the *man pages* (On-Line Manual)

- Use `man` to see man pages on a named command, e.g.


```
$ man date
```
- The result should be something like:

```
DATE(1)                                FSF                                     D

NAME
    date - print or set the system date and time

SYNOPSIS
    date [OPTION]... [+FORMAT]
    date [-u|--utc|--universal] [MMDDhhmm[[CC]YY][.ss]]
```

- `DATE(1)` Shows page is in manual section 1
- To view a page from a certain section use:


```
$ man -S section-number command
```