Chapter 1

Introduction to UNIX Operating System

An operating system is a software that perform many of the low-level tasks; for example allocating storage, scheduling tasks and handing the peripheral hardwares. The peripheral hardware are, for example, disk drives, printers, mouse, the screen, and keyboard. An operating system has two main parts: the *kernel* and the *system programs*. The kernel allocates machine resources–including memory, disk space, and *CPU* cycles–to all other programs that run on the computer. The system programs include shells (command interpreters), configuration scripts and files, device drivers, utility programs, libraries, application programs, servers, and documentations. Many of the libraries, servers, and utility programs were written by the GNU Project.

1.1 Overview of Linux

The Linux operating system has many unique features. For example, it is a control program for computers like other other operating systems. However, being UNIX-like, it also contains a concordant set of utility programs that could work together efficiently and a set of tools that allow users to connect and use these utilities to build systems and applications. Some of the unique and powerful features of the Linux operating system are

Linux Has a Kernel Programming Interface

The Linux kernel—the heart of the Linux operating system—is responsible for allocating the computer's resources and scheduling user jobs so each one gets its fair share of system resources, including access to the CPU; peripheral devices, such as hard disk, DVD, and CD-ROM storage; printers' and tape drives. Programs interact with the kernel through *system calls*, special functions with well-known names. A programmer can use a single system call to interact with many kinds of devices. For example, there is one **write**() system call, rather than many device-specific ones. When a program issues a **write**() request, the kernel interprets the context and passes the request to the appropriate device. This flexibility allows old utilities to work with devices that did not exist when the utilities were written.

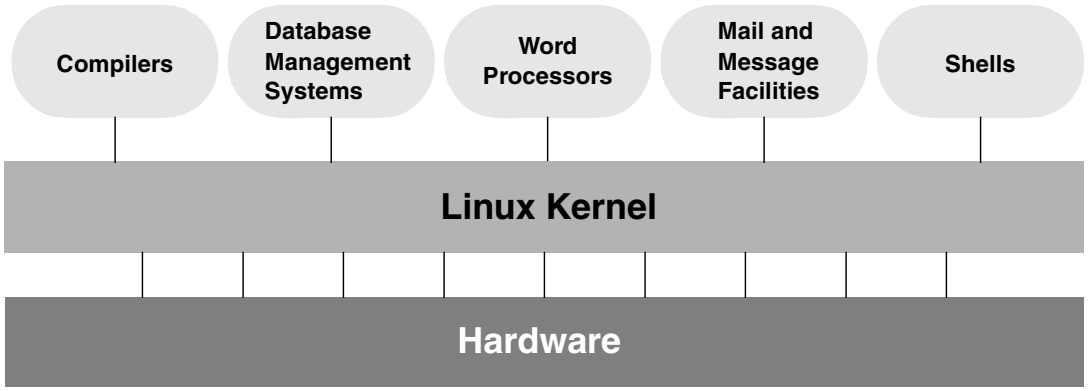

Figure 1.1: A layered view of the Linux operating system.

It also makes it possible to move programs to new versions of the operating system without rewriting them.

Linux Can Support Many Users

Depending on the hardware and the types of tasks the computer performs, a Linux system can support from 1 to more than 1,000 users, each concurrently running a different set of programs. The per-user cost of a computer that can be used by many people at the same time is less than that of a computer that can be used by only a single person at a time. It is less because one person cannot generally take advantage of all the resources a computer has to offer. That is, no one can keep all the printers going constantly, keep all the system memory in use, keep all the terminals busy at the same time. By contrast, a multiuser operating system allows many people to use all of the system resources almost simultaneously. The use of costly resources can be maximized and the cost per user can be minimized—the primary objectives of a multiuser operating system.

Linux Can Run Many Tasks

Linux is a fully protected multitasking operating system, allowing each user to run more than one job at a time. Processes can communicate with one another but remain fully protected from one another, just as the kernel remains protected from all processes. We can run several jobs in the background while giving all our attention to the job being displayed on the screen, and we can switch back and forth between jobs. This capability helps users be more productive.

Linux Provides a Secure Hierarchical Filesystem

A *file* is a collection of information, such as text for a memo or report, an accumulation of sales figures, an image, a song, or an executable program. Each file is stored under unique identifier on

a storage device, such as a hard disk. The Linux filesystem provides a structure whereby files are arranged under *directories*, which are like folders or boxes. Each directory has a name and can hold other files and directories. Directories, in turn, are arranged under other directories, and so forth, in a treelike organization. This structure helps users keep track of large numbers of files by grouping related files in directories. Each user has one primary directory and as many subdirectories as required (Figure. 1.2).

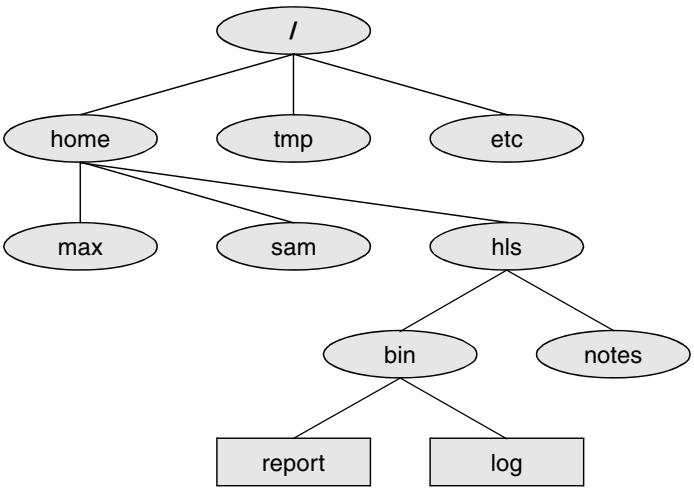

Figure 1.2: The Linux filesystem structure.

A *link* allows a given file to be accessed by means of two or more names. The alternative names can be located in the same directory as the original file or in another directory. Links can make the same file appear in several users' directories, enabling those users to share the file easily. Windows used the term *shortcut* in place of *link* to describe this capability. Macintosh users will be more familiar with the term *alias*. Under Linux, an *alias* is different from a *link*; it is a command macro feature provided by the shell.

1.2 The Utilities

When Linux was first introduced, it ran using a textual interface. All the tools ran from a command line. Today the Linux has a graphical user interface (GUI). However, command-line utilities are often faster, more powerful, or more complete than their GUI counterparts. Sometimes there is no GUI counterpart to a textual utility. When we work with a command-line interface, we are working with a shell. Hence, it is important that we understand something about the characters that are special to the shell.

1.2.1 Special Characters

Special characters, have a special meaning to the shell. These characters are mentioned here so we should avoid accidentally using them as regular characters until we understand how the shell interprets them.

```
& ; | * ? ' " ` [ ] ( ) $ < > { } # / \ ! ~
```

Although not considered special characters, `RETURN`, `SPACE`, and `TAB` have special meanings to the shell. For example, `RETURN` usually ends a command line and initiates execution of a command. The `SPACE` and `TAB` characters separate elements on the command line and are collectively known as *whitespace* or *blanks*.

If we need to use a character that has a special meaning to the shell as a regular character, we can *quote* (or *escape*) it. When we quote a special character, we keep the shell from giving it special meaning. The shell treats a quoted special character as a regular character. However, a slash (`/`) is always a separator in a pathname, even when we quote it.

1.2.2 Basic Utilities

One of the important advantage of UNIX is that it comes with thousands of utilities that perform countless functions. We will use utilities whenever we work with Linux, whether we use them directly by name from the command line or indirectly from a menu or icon. When we work with a command-line interface we are working with a shell (command interpreters).

The term *directory* is used extensively in the next sections. A directory is a resource that can hold files. On other operating system, including Windows and Mac OS X, and frequently when when speaking about a Linux GUI, a directory is referred to as a *folder*.

The most commonly used commands are shown in Table 1.1. The following sections, we shall describe some of the basic commands in more detail.

1.2.3 Working with Files and Directories

We shall start learning Linux command in a tutorial style where you can type the command and see the result immediately. This section describes utilities that copy, move, print, search through, display, sort and compare files.

Filename completion

> After you enter one or more letters of a filename (following a command) on a command line, press `TAB` and the Bourne Again Shell will complete as much of the filename as it can. When only one filename starts with the characters you entered, the shell completes the filename and place a `SPACE` after it. You can keep typing or you can press `RETURN` to execute the command at this point.

Command	Descriptions
`apropos`	Search the database for strings.
`cat`	Display file's contents to the standard output device.
`cd`	Change directory.
`chmod`	Change a file's permissions.
`clear`	Clear a command line screen/window for a fresh start.
`cp`	Copy files and directories.
`date`	Display or set the system date and time.
`df`	Display used and available disk space.
`du`	Show how much space each file takes up.
`file`	Determine what type of data is within a file.
`grep`	Search files or output for a particular pattern.
`head`	Display the first n lines of a file (the default is 10).
`kill`	Stop a process.
`less`	View the contents of a file one page at a time.
`ln`	Create a shortcut.
`locate`	Search a copy of your filesystem for the specified filename.
`ls`	List directory contents.
`man`	Display the help information for the specified command.
`mkdir`	Create a new directory.
`mv`	Rename or move file(s) or directories.
`passwd`	Change the password or allow (for the system administrator) to change any password.
`ps`	Display a snapshot of the currently running processes.
`pwd`	Display the pathname for the current directory.
`rm`	Remove (delete) file(s) and/or directories.
`rmdir`	Delete empty directories.
`ssh`	Remotely log in to another Linux machine, over the network. Leave an ssh session by typing **exit**.
`sudo`	Execute a command as another user
`tail`	Display the last n lines of a file (the default is 10).
`tar`	Manipulate tape archives.
`top`	Create an empty file with the specified name.
`touch`	Create an empty file with the specified name.
`w`	Display who is logged in and what they are doing.

Table 1.1: Descriptions of basic Linux commands that are commonly used.

`pwd`: Return Working Directory Name

The `pwd` utility writes the absolute pathname of the current working directory to the standard output. Try typing the following command and see the result – to execute the command type the command and then press `ENTER`.

```
[Leon:~]$ pwd
/Users/Teeraparb
[Leon:~]$
```

The command will return the full path of your current working directory. Initially, you will start with your home directory. To see the content of your home directory type the command `ls`.

`ls`: List the Names of Files

The `ls` utility display information about one or more files. It lists the information alphabetically by filename unless you use an option that changes the order.

```
[Leon:~]$ ls
Application                     Google Drive                    Sites
Applications                    Library                         Source
Applications (Parallels)        Movies                          Textbooks
Desktop                         Music                           Work
Documents                       Pictures                        bin
Downloads                       Projects                        lib
Dropbox                         Public                          tmp
[Leon:~]$
```

The optional arguments can modified the outcome of the utility. For example, `-l` lists more information about each file.

```
[Leon:~]$ ls -l
total 0
drwxr-xr-x  16 Teeraparb  staff   544 Nov  6 00:26 Application
drwxr-xr-x   6 Teeraparb  staff   204 Jan 10  2016 Applications
drwxr-xr-x@  6 Teeraparb  staff   204 Mar  7  2016 Applications (Parallels)
drwx------+ 24 Teeraparb  staff   816 Jan  5 09:17 Desktop
drwx------+ 25 Teeraparb  staff   850 Nov 15 10:00 Documents
drwx------+ 31 Teeraparb  staff  1054 Jan  1 18:27 Downloads
drwx------@ 19 Teeraparb  staff   646 Dec 26 09:59 Dropbox
drwx------@ 13 Teeraparb  staff   442 Dec 26 09:59 Google Drive
drwx------@ 71 Teeraparb  staff  2414 Jun  1  2016 Library
drwx------+ 11 Teeraparb  staff   374 Jun  1  2016 Movies
drwx------+  5 Teeraparb  staff   170 Mar 16  2013 Music
drwx------+ 14 Teeraparb  staff   476 May 30  2016 Pictures
drwx------  27 Teeraparb  staff   918 Oct 24 08:14 Projects
drwxr-xr-x+  6 Teeraparb  staff   204 Feb  2  2016 Public
drwxr-xr-x+  7 Teeraparb  staff   238 Apr 19  2014 Sites
drwxr-xr-x  22 Teeraparb  staff   748 Nov 30  2015 Source
drwxr-xr-x  10 Teeraparb  staff   340 Feb 23  2016 Textbooks
drwxr-xr-x  11 Teeraparb  staff   374 Feb 15  2016 Work
drwxr-xr-x   8 Teeraparb  staff   272 Jun 30  2015 bin
drwxr-xr-x   6 Teeraparb  staff   204 Mar 13  2016 lib
drwx------  19 Teeraparb  staff   646 May 24  2016 tmp
[Leon:~]$
```

The optional argument `-t` will sorted the files by the time they were last modified. When added to `-l` will give the details sorted by time. The optional arguments can be stacked to get combined effect, for example.

```
[Leon:~]$ ls -lt
total 0
drwx------+ 24 Teeraparb  staff   816 Jan  5 11:20 Desktop
drwx------+ 31 Teeraparb  staff  1054 Jan  1 18:27 Downloads
drwx------@ 13 Teeraparb  staff   442 Dec 26 09:59 Google Drive
drwx------@ 19 Teeraparb  staff   646 Dec 26 09:59 Dropbox
drwx------+ 25 Teeraparb  staff   850 Nov 15 10:00 Documents
drwxr-xr-x  16 Teeraparb  staff   544 Nov  6 00:26 Application
drwx------  27 Teeraparb  staff   918 Oct 24 08:14 Projects
drwx------@ 71 Teeraparb  staff  2414 Jun  1  2016 Library
drwx------+ 11 Teeraparb  staff   374 Jun  1  2016 Movies
drwx------+ 14 Teeraparb  staff   476 May 30  2016 Pictures
drwx------  19 Teeraparb  staff   646 May 24  2016 tmp
drwxr-xr-x   6 Teeraparb  staff   204 Mar 13  2016 lib
drwxr-xr-x@  6 Teeraparb  staff   204 Mar  7  2016 Applications (Parallels)
drwxr-xr-x  10 Teeraparb  staff   340 Feb 23  2016 Textbooks
drwxr-xr-x  11 Teeraparb  staff   374 Feb 15  2016 Work
drwxr-xr-x+  6 Teeraparb  staff   204 Feb  2  2016 Public
drwxr-xr-x   6 Teeraparb  staff   204 Jan 10  2016 Applications
drwxr-xr-x  22 Teeraparb  staff   748 Nov 30  2015 Source
drwxr-xr-x   8 Teeraparb  staff   272 Jun 30  2015 bin
drwxr-xr-x+  7 Teeraparb  staff   238 Apr 19  2014 Sites
drwx------+  5 Teeraparb  staff   170 Mar 16  2013 Music
[Leon:~]$
```

Since we are going through the tutorial, it is better to create our own directory.

`mkdir`: Creates a Directory

The `mkdir` utility creates a directory. The *argument* to `mkdir` becomes the pathname of the new directory. The following examples develop the directory. Now execute the command

```
[Leon:~]$ mkdir tutorial
[Leon:~]$
```

Now we have a new working directory *tutorial*. Let see if the directory already exists.

```
[Leon:~]$ ls -ltr
total 0
drwx------+  5 Teeraparb  staff   170 Mar 16  2013 Music
drwxr-xr-x+  7 Teeraparb  staff   238 Apr 19  2014 Sites
drwxr-xr-x   8 Teeraparb  staff   272 Jun 30  2015 bin
drwxr-xr-x  22 Teeraparb  staff   748 Nov 30  2015 Source
drwxr-xr-x   6 Teeraparb  staff   204 Jan 10  2016 Applications
drwxr-xr-x+  6 Teeraparb  staff   204 Feb  2  2016 Public
drwxr-xr-x  11 Teeraparb  staff   374 Feb 15  2016 Work
drwxr-xr-x  10 Teeraparb  staff   340 Feb 23  2016 Textbooks
drwxr-xr-x@  6 Teeraparb  staff   204 Mar  7  2016 Applications (Parallels)
drwxr-xr-x   6 Teeraparb  staff   204 Mar 13  2016 lib
drwx------  19 Teeraparb  staff   646 May 24  2016 tmp
drwx------+ 14 Teeraparb  staff   476 May 30  2016 Pictures
drwx------+ 11 Teeraparb  staff   374 Jun  1  2016 Movies
drwx------@ 71 Teeraparb  staff  2414 Jun  1  2016 Library
drwx------  27 Teeraparb  staff   918 Oct 24 08:14 Projects
drwxr-xr-x  16 Teeraparb  staff   544 Nov  6 00:26 Application
drwx------+ 25 Teeraparb  staff   850 Nov 15 10:00 Documents
drwx------@ 19 Teeraparb  staff   646 Dec 26 09:59 Dropbox
drwx------@ 13 Teeraparb  staff   442 Dec 26 09:59 Google Drive
drwx------+ 31 Teeraparb  staff  1054 Jan  1 18:27 Downloads
drwx------+ 26 Teeraparb  staff   884 Jan  5 11:49 Desktop
drwxr-xr-x   2 Teeraparb  staff    68 Jan  5 11:55 tutorial
[Leon:~]$
```

The command `ls -ltr` will give you the details of the contents in the directory as well as sort them by the modified time stamp. The optional argument `-r` will reverse the order of the display. Now we will move to the new directory.

`cd`: Change to Another Working Directory

The `cd` (change directory) utility makes another directory the working directory.

```
[Leon:~]$ cd tutorial/
[Leon:tutorial]$
```

Now type the command `pwd` again, and you should be in the directory `tutorial`. Now we can going to create a file so that you can do something with it.

`cat`: Concatenate and Print Files

The `cat` utility reads files sequentially, writing them to the standard output. The utility displays the contents of a text file. The name of the command is derived from *concatenate*, which means to join together, one after the other.

```
[Leon:tutorial]$ cat > practice
This is the first line.
This is the second line.
Type ctr-c to end the input.
^C
[Leon:tutorial]$
```

Now you should have the file name `practice` in the directory. Check with the command `ls`. The argument of the `cat` utility is an example of input/output redirection which we will cover later in section 1.4

`touch`: Create a File or Change Modification Time

The `touch` utility changes the access and/or modification time of a file to the current time or a time you specify. You can also use `touch` to create a file.

```
[Leon:tutorial]$ touch practice
[Leon:tutorial]$
```

Now the modification time stamp of the file `practice` should be updated. In addition, the utilities `touch` can create a blank file (a file without any content).

```
[Leon:tutorial]$ touch blankfile
[Leon:tutorial]$
```

`less`: Display a Text File One Screen at a Time

When you want to view a file that is longer than one screen, you can use the `less` utility. This utility pauses after displaying a screen of text; press the `SPACE` bar to display the next screen of text. Because this utility show one page at a time, it is called *pager*.

```
[Leon:tutorial]$ less practice
[Leon:tutorial]$
This is the first line.
This is the second line.
Type ctr-c to end the input.
practice lines 1-3/3 (END)
```

In order to, exit the display mode of the utility `less` type `q`. Table 1.2 shows all the command in display mode of `less` utility.

Command	Movement
e or j or RETURN	Forward one line
y or k	Backward one line
f or z or SPACE	Forward one window
b or w	Backward one window
d	Forward one half window
u	Backward one half window
h	Display help window
q	Exit

Table 1.2: The command in display mode for the utilities `less` and `man`

`cp`: Copies Files

The `cp` utility copies one or more files. It can either make a copy of a single file or copy one or more files to a directory. This utility can copy any file, including text and executable program (binary) file. You can use `cp` to make a backup copy of a file or a copy to experiment with.

The `cp` command line uses the following syntax to specify source and destination files:

*cp **source-file destination-file***

The ***source-file*** is the name of the file that `cp` will copy. The ***destination-file*** is the name of that `cp` assigns to the resulting (new) copy of the file.

```
[Leon:tutorial]$ cp practice practice2
[Leon:tutorial]$
```

You can check the contains of the file `practice` and `practice2` that they are the same.

`mv`: Changes the Name of Files

The `mv` (move) utility can rename a file without making a copy of it. The `mv` command line specified an existing file and a new filename using the same syntax as `cp`:

*mv **existing-file new-filename***

```
[Leon:tutorial]$ mv practice practice3
[Leon:tutorial]$
```

Now we can check if the file has been renamed to `practice3`

`rm`: Deletes Files

The `rm` (remove) utility deletes a file. After `rm` deletes the file, `ls` and `cat` show that `practice2` is no longer in the directory. The `ls` utility does not list its filename, and `cat` says that no such file

exists. Use `rm` carefully.

```
[Leon:tutorial]$ rm -f practice2
[Leon:tutorial]$
```

1.3 Where to Find Document

Distribution of Linux do not typically come with hardcopy reference manuals. However, its online documentation has always been one of Linux's strengths. The man (or manual) and info pages have been available via the `man` and `info` utilities since early releases of the operating system.

`man`: Display the System Manual

The `man` utility display (man) pages from the system documentation in a textual environment. This documentation is helpful when we know which utility we want to use but have forgotten exactly how to use it. We can also refer to the `man` page to get more information about specific topics or to determine which features are available with Linux.

```
[Leon:tutorial]$ man less
LESS(1)                                                         LESS(1)

NAME
       less - opposite of more

SYNOPSIS
       less -?
       less --help
       less -V
       less --version
       less [-[+]aABcCdeEfFgGiIJKLmMnNqQrRsSuUVwWX~]
            [-b space] [-h lines] [-j line] [-k keyfile]
            [-{oO} logfile] [-p pattern] [-P prompt] [-t tag]
            [-T tagsfile] [-x tab,...] [-y lines] [-[z] lines]
            [-# shift] [+[+]cmd] [--] [filename]...
       (See the OPTIONS section for alternate option syntax with long option names.)

DESCRIPTION
       Less  is  a  program similar to more (1), but which allows backward movement in
       the file as well as forward movement.  Also, less does not  have  to  read  the
       entire  input  file  before  starting,  so  with large input files it starts up
       faster than text editors like vi (1).  Less uses termcap (or terminfo  on  some
       systems),  so it can run on a variety of terminals.  There is even limited sup-
       port for hardcopy terminals.  (On a hardcopy terminal, lines  which  should  be
       printed at the top of the screen are prefixed with a caret.)

       Commands  are based on both more and vi.  Commands may be preceded by a decimal
       number, called N in the descriptions below.  The number is used  by  some  com-
       mands, as indicated.
:
```

The man utility use a set of commands to move the screen display as less command. Table 1.2 shows all the command in display mode of more utility.

apropos: Searches for a Keyword

When we do not know the name of the command we need to carry out a particular task, we can use apropos with a keyword to search for it. This utility searches for the keyword in the short description line (the top line) of all man pages and displays those that contain a match. The man utility, when called with the -k (keyword) option, provides the same output as apropos.

```
[Leon:tutorial]$ apropos who
gifspnge(1)              - expensive GIF copy, whole-image version
Net::LDAP::Extension::WhoAmI(3pm) - LDAP "Who am I?" Operation
biff(1)                  - be notified if mail arrives and who it is from
from(1)                  - print names of those who have sent mail
ldapwhoami(1)            - LDAP who am i? tool
rwho(1)                  - who is logged in on local machines
rwhod(8)                 - system status server
w(1)                     - display who is logged in and what they are doing
who(1)                   - display who is logged in
whoami(1)                - display effective user id
whois(1)                 - Internet domain name and network number directory service
[Leon:tutorial]$
```

The apropos searches a set of database files containing short descriptions of system commands for keywords and displays the result on the standard output.

1.4 I/O Redirection

We will explore a powerful feature used by many command line programs called input/output redirection. As we have seen, many commands such as ls print their output on the display. This does not have to be the case, however. By using some special notation we can redirect the output of many commands to files, devices, and even to the input of other commands.

Standard Output

Most command line programs that display their results do so by sending their results to a facility called standard output. By default, standard output directs its contents to the display. To redirect standard output to a file, the > character is used like this:

```
[Leon:tutorial]$ ls > file_list.txt
[Leon:tutorial]$
```

In this example, the ls command is executed and the results are written in a file named file_list.txt. Since the output of ls was redirected to the file, no results appear on the display.

Each time the command above is repeated, `file_list.txt` is overwritten (from the beginning) with the output of the command ls. If we want the new results to be appended to the file instead, use >> like this:

```
[Leon:tutorial]$ ls >> file_list.txt
[Leon:tutorial]$
```

When the results are appended, the new results are added to the end of the file, thus making the file longer each time the command is repeated. If the file does not exist when we attempt to append the redirected output, the file will be created.

Standard Input

Many commands can accept input from a facility called standard input. By default, standard input gets its contents from the keyboard, but like standard output, it can be redirected. To redirect standard input from a file instead of the keyboard, the < character is used like this:

```
[Leon:tutorial]$ sort < file_list.txt
blankfile
blankfile
file_list.txt
file_list.txt
practice3
practice3
[Leon:tutorial]$
```

The results are output on the display since the standard output is not redirected in this example. We could redirect standard output to another file like this:

```
[Leon:tutorial]$ sort < file_list.txt > sorted_file_list.txt
[Leon:tutorial]$
```

As we can see, a command can have both its input and output redirected. Be aware that the order of the redirection does not matter. The only requirement is that the redirection operators (the < and >) must appear after the other options and arguments in the command.

Pipelines

By far, the most useful and powerful thing we can do with I/O redirection is to connect multiple commands together with what are called pipes. With pipes, the standard output of one command is fed into the standard input of another. For example:

```
[Leon:tutorial]$ ls -lt | head -2
total 24
-rw-r--r--  1 Teeraparb  staff  68 Jan  8 16:17 sorted_file_list.txt
[Leon:tutorial]$
```

In this example, the output of the `ls -lt` command is fed into `head -2`, which will display the

first 2 lines of the input. The resulting command gives you the most recent file updated. By connecting commands together, we can accomplish amazing feats.

References

- Arnold Robbins, “UNIX in a Nutshell”, 4th Edition, O’Reilly Media, 2005.
- Jerry Peek, Grace Todino, and John Strang, “Learning the Unix Operating System”, 5th Edition, O’Reilly Media, 2002.

Homework

1. Your first assignment is to install the Linux operating system on your computer, or use one of the Linux-installed computers in the institute.

   (a) Find out what is the shell your Linux computer is using.

   (b) Find out what is the IP-address of the computer.

   Write down the commands.

2. The `ls` command will be one of your most useful command. Find out how to we use the command so that we can distinguish directories, files, links and executable file-types.

3. The `rm`, `mv` and `cp` are dangerous commands as it can permanently delete files in your machine. How do we prevent unintentionally delete files using the commands? Think about, for example, typing the command `rm myfile` then press `ENTER`.

4. What are these commands doing? Describe.

   (a) `cat test1 test2 > test3`

   (b) `wc -l < barry.txt > myoutput`

   (c) `ls | head -3 | tail -1`

5. Download the file from the link below and sort the data in the sixth column by numerical order. Write down the command that you use.
   Then send your result to my E-mail: `teeraparc@nu.ac.th`
   `https://drive.google.com/file/d/0B_89najODSgCZFZTSTVWUVBIUlk/view?usp=sharing`
   You may need to search for an appropriate Linux command to do this task.