



~~~~~ A Brief History And Introduction ~~~~~

Richard Stallman: GNU (GNU's Not Unix) project founder (Launched in 1984 to develop a complete Unix-like operating system); Also the founder of Free Software Foundation (FSF); His creations: Emacs Editor, GNU C Compiler (gcc), ...

Linus Torvald: Author of the Linux; At beginning for 386 and 486 architectures; Written by gcc;
About penguin sign of Linux...

Why do they like Linux?

- ✓ Linux is Open Source (See kernel.org & sourceforge.net.)
- ✓ Linux is Secure (See www.linuxsecurity.com)
- ✓ Linux is Simple, Clear and Beautiful
- ✓ Linux is Free
- ✓ Linux is very Flexible
- ✓ Linux is a Strong and Stable OS
- ✓ Linux has many Forums, Resources, ... (See linuxquestions.org)
- ✓ Linux has fast Development and Versioning (See distrowatch.com)



Why do they hate Linux?

- ✓ Linux is Wizard less
- ✓ People doesn't know its Softwares
- ✓ People can not find needed Softwares
- ✓ **They are Accustomed to Windows**



~~~~~ Basic Topics About Linux ~~~~~

All things here is Open Source (See <http://www.kernel.org> & <http://www.sourceforge.net>.)

Text & Graphical Environment. (Alt + Ctrl + F1...F7) (Alt + Left / Right Arrow key)

Windows Managers: KDE, GNOME, ...

root = Administrator

/ = My Computer

Almost all things in Linux is a file

FHS (File System Hierarchy Standard)

- ✓ / The root directory
- ✓ /boot Files of the boot loader
- ✓ /home Users home (like Document and Setting's in Windows)
- ✓ /root root's home
- ✓ /dev Device files
- ✓ /etc System configuration's file
- ✓ /lib Shared libraries
- ✓ /mnt Suggested mount point for mounting a file system temporarily
- ✓ /bin Simple binary files

- ✓ /sbin root's binaries
- ✓ /tmp Temporary files



- ✓ /usr Secondary hierarchy
- ✓ ...

Each user has a home (cd ~ / cd)

Hidden files in Linux begins with “.”; i.e. a file is hidden if and only if its name begin with “.”

Alt + F2 = Win key + R (in Windows) : open run window

~~~~~ **Using Shell** ~~~~~

Linux is case-sensitive

Most of Linux commands are lowercase

Manual pages:

1: User commands (executable programs or shell commands)

2: System calls (functions provided by the kernel)

3: Programmers manual (functions within program libraries)

...

8: root's command

Commands:

- ✓ clear
- ✓ yppasswd
- ✓ ls
- ✓ pwd
- ✓ cd
- ✓ mkdir
- ✓ rmdir
- ✓ rm
- ✓ cp (Example: cp /etc/shadow .-> Permission denied)
- ✓ cat
- ✓ top
- ✓ ps (-A -a)
- ✓ logout = Ctrl + d
- ✓ mount
 - /dev/hda1 c
 - /dev/hda5 d
- ✓ ...

Tab usage: Auto complete; Never forgot it! (Even in su, Makefile, ...)

clear = Ctrl + l

Shift + Page Up / Page Down: Screen scroll

Commands history: Up / Down Arrow key (.bash_history)

An important hint: Enter character(s): Windows: “\r\n” Linux: “\n” Macintosh: “\r” (dos2unix command)

~~~~~ **mc (Midnight Commander)** ~~~~~

A very simple and useful file manager

Even opens zipped / tar / ... files

Ctrl + s

Search: Esc + ?



~~~~~ vi ~~~~~

A wonderful text editor!

It knows almost all file types! (C, C++, HTML, Java, php, Shell Script, most configuration files, ...)

vim fileName +10 (a number) -> Open the file with name fileName and go to the line 10

i: Go to insert mode

dd: Delete a line

u: Undo

q: Quit

q!: Quit without saving

w: Save

/: Search

Ctrl + p: Auto complete (Wonderful! If you include some file in your C program, it will consider them too!)

~~~~~ Useful Information About UT Linux System ~~~~~

Don't have a linux account still?!! <http://khorshid.ut.ac.ir/signup.html>

Even you can access your linux account, in MS Windows by ftp or SSH... <ftp://khorshid.ut.ac.ir>

For SSH you can use putty.exe (<http://khorshid.ut.ac.ir/userguides/putty.exe>)

~~~~~ Programming Under Linux ~~~~~

See a "Hello World" Example ...

main is the start point of the program

Style & Indentation

Give your variable GOOD LONG names, Don't worry about typing their name

About pointers (Every array name in C is a pointer to the first of it)

Making a binary file phases:

1: Preprocessor

2: Compiler

3: Linker

Usual switches:

- ✓ -c Only compile the program
- ✓ -o Output file name
- ✓ -l Use this library

At the compile time, prototype of the functions should be clear

You can not write your programs always in a simple file (See an example about handling two files)

Makefile (You may use my bash script, for writing your Makefiles)

~~~~~ Multi Thread Programming ~~~~~

Time and duration of each thread is handled by OS; You can not think of a special order for it

Shared datas should not be accessed with more than a thread at a time; So if a memory location may be accessed by more than a thread for write, you should use mutex to access that...

pthread_create (a pointer to a pthread_t variable, a pointer to a thread attribute object, a pointer to the thread function, a thread argument value of type void*)

pthread_exit (the thread ID which we want to exit)



pthread_join (the thread ID which we want to wait for it to finish, a pointer to a void* variable to save the return value of the thread)

Thread function should have the form (void*) f (void*)

~~~~~ Socket Programming ~~~~~

Socket: a way to speak to other programs using standard Unix file descriptors

There are some socket types; We will discuss just Internet Sockets (TCP/IP);

We have two kinds of Internet Sockets: Stream Sockets (SOCK_STREAM or connectionless sockets), Datagram Sockets (SOCK_DGRAM). Stream sockets are reliable two-way connected communication streams. If you output two items into the socket in the order "1, 2", they will arrive in the order "1, 2" at the opposite end. They will also be error free. HTTP protocol uses this sockets. It is based on TCP. but how about Datagram sockets? if you send a datagram, it may arrive. It may arrive out of order. If it arrives, the data within the packet will be error-free. they use the "User Datagram Protocol", or "UDP". You just build a packet, slap an IP header on it with destination information, and send it out. No connection needed. They are generally used for packet-by-packet transfers of information. Application that uses this, has it's own protocol on top of UDP. For example, the tftp protocol says that for each packet that gets sent, the recipient has to send back a packet that says, "I got it!" (an "ACK" packet.)

Now lets see a simple chat program with Stream sockets...

~~~~~ Linux Softwares ~~~~~

See <http://linuxshop.ru/linuxbegin/win-lin-soft-en/table.shtml>

IDE: *anjuta, kdevelop, eclipse, rhide, ...*

Text Editor: *kate, kedit, gvim, kwrite, emacs, ...*

Messenger: *gaim, licq, kopete, ...*

Web Browser: *mozilla, konqueror, opera, firebird, ...*

Multi Media: *mplayer, xmms, winamp, noatun, xine, ...*

CD Writer: *k3b, arson, gcombust, simplecdr-x, ...*

Download Manager: *wget, prozilla, aria, lftp, ...*

PDF Reader: *acrobat reader, xpdf, gv, ghostview, ...*

Office Tools: *open office, koffice, staroffice, ...*

Photo Manager (Viewer, Editor, ...): *kpaint, gimp, xnview, imagemagick, ...*

Wonderfuls: *wine, bb, ...*

LOTS OF GAMES!!!

...

~~~~~ Structured Programming & Object Oriented Programming ~~~~~

In the Structured Programming we had functions and structures; We passed a structure to a function,

--The programmer himself is responsible for the relations between functions and structures;

--Isn't it better that the programming language handles this?!!

--Data security is low; A global variable is reachable from all the point of the program and can be modify;

--Isn't it better that every part of the program can only access a variable that it really needs it?!!

--Structured Programming itself is not real; Nothing is like the real world;

--Isn't it better that we make our code, the same as the real word?!!



What do we have in the world?!! "Objects"

Real worlds is made of "Object"s; Each "Object" has some properties (Data members); Each "Object" has interfaces that get the other "Object" a way to communicate to it (member functions);

OO provides:

- Abstraction
- Encapsulation
- Data Hiding

Class Student

```
{
    private:                                --> private Data Members

        string ID;
        string name;
        list <Course> currentSemesterCourses;

    public:                                  --> These are Public Members

        Student (string anID, string aName);    --> Constructor
        ~Student ();                            --> Destructor

        string getName ();                     --> a Getter Method
        list <Course> getCourses ();

        void setName (string newName);        --> a Setter Method
};
```

Now we say ...

```
Student s1 ("203", "Hamid Hajabdolali");
s1.getName ();
```

.
.
.

```
s1.getCourses (); (Isn't it better than getCourse (&s1); ?!!)
```

For the backward compatibility, C++ let the programmer writes his codes in OO or like C (Structured).
See some Examples ...

~~~~~ QT ~~~~~

A very famous GUI Toolkit used to make KDE; Maintained by Trolltech; cross platform; (About KDE; It is an internet project based on QT; It was founded by a mail! In 1996, Matthias Ettrich criticizes the poor graphical environments on Unix/Linux! Claimed that CDE, X Windows are no desktop environment comparing to one for MS Windows; Asked for developers to gather and work on a new free Kool Desktop Environment (KDE)) Contributors include hundreds of developers around the world. Many projects sprung on KDE including Kmail, KDevelop, KOffice, Konquerer, and it went worldwide...



## ~~~~~ IPC (Inter Process Communication) ~~~~~

In simple words it is a simple mechanism which OS supports for process to communicate to each other;  
See the simplest example I could think of...

~~~~~ Any Questions?!! ~~~~~

hhamid@gmail.com

~~~~~ Other Useful Links ~~~~~

- www.advancedlinuxprogramming.com
- www.linuxdoc.org
- <http://docs.linux.cz/>
- www.linuxsoft.cz

