



**The Abdus Salam
International Centre for Theoretical Physics**



1967-3

Advanced School in High Performance and GRID Computing

3 - 14 November 2008

Introduction to LINUX

KOHLMEYER Axel and JOHNSON Robert R.

*University of Pennsylvania
Department of Chemistry
231 South 34th Street
PA 19104 Philadelphia
U.S.A.*

Introduction to Linux

Text mode interface

Robert R. Johnson, Axel Kohlmeyer

University of Pennsylvania

Philadelphia, USA



the **abdus salam**
international centre for theoretical physics



Outline

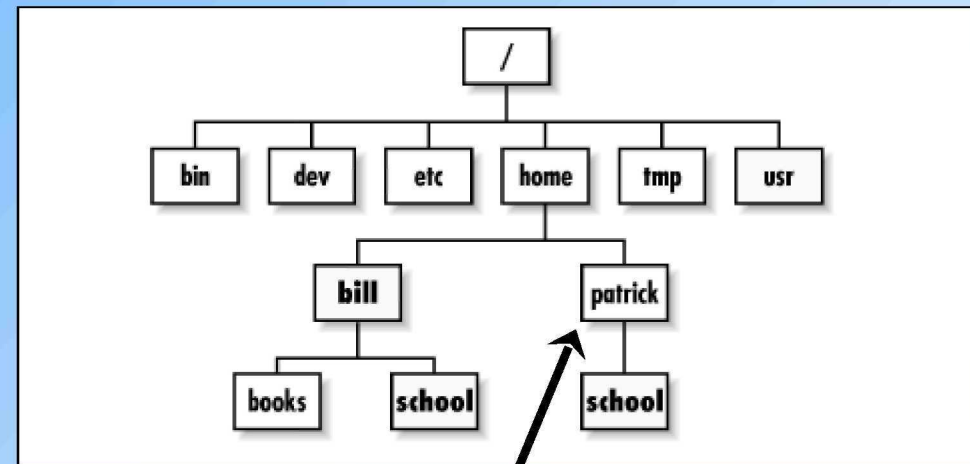
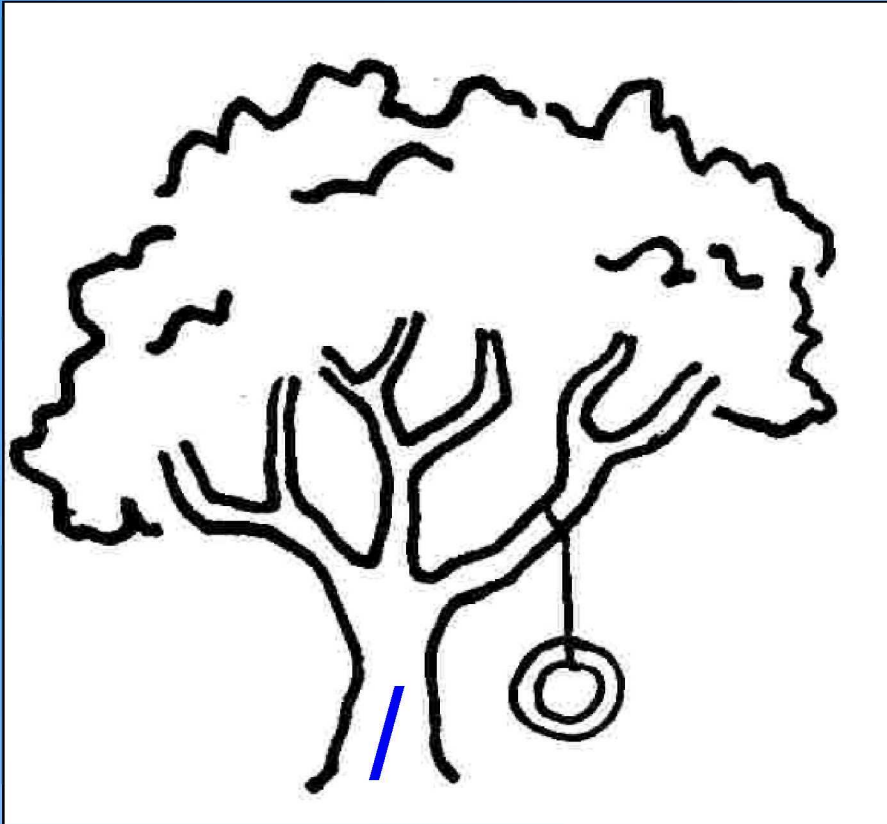
- Preliminaries
- Basic Commands
- Combining/Redirecting
- Environment Variables
- Scripts
- Loops, Conditions
- vi text editor



“Linux is user-friendly. It is not ignorant-friendly or idiot-friendly.”

Preliminaries – File Organization

Directory Tree



Patrick's Home Directory

`/home/patrick`

Preliminaries – The Shell

```
Terminal
File Edit View Terminal Tabs Help
[bob@nickel ~]$ cowsay -f elephant Hi! This is the shell!
-----
< Hi! This is the shell! >
-----
  \      ^__^
  (oo)\_______
      (__)\       )\/\
         ||----w |
         ||     ||
         [m]   [m]
[bob@nickel ~]$
```

Basic Linux Commands

Commands:

- What are they?
Just files in the directory tree
- Where are they located?
`/bin, /usr/bin, /sbin`
- How do I determine how a command works?
Man pages: `man command-name`
- Anatomy of a command:
`[DoSomething] [How] [ToFiles/Directories]`
`ls -l /home/bob`

Basic Linux Commands

Basic Navigation

- ls (ls -ltr)
- pwd
- cd

Directory Creation

- mkdir
- rmdir

File Viewing

- more, less
- head, tail
- grep, wc

File Manipulation

- cp
- mv
- rm
- touch
- rename
- cat, paste
- chmod, chgrp, chown

System Information

- top, ps
- kill
- du, df

Controlling Data Flow

■ Redirection

- Redirect output into a new file: '>'
command > filename
- Append output to an existing file: '>>'
command >> filename
- Direct file as input for command: '<'
command < input_file

■ Piping

- Use output from one command as input for a second: '|'
command1 | command2 | command3...

Shell Variables

Store numbers, filenames, strings in variable that's accessible to the shell

Local Variables

Variable definition: `VAR=value`

Obtain value of variable: `${VAR}`

Print value of variable: `echo ${VAR}`

Unset variable: `unset VAR`

Environment Variables

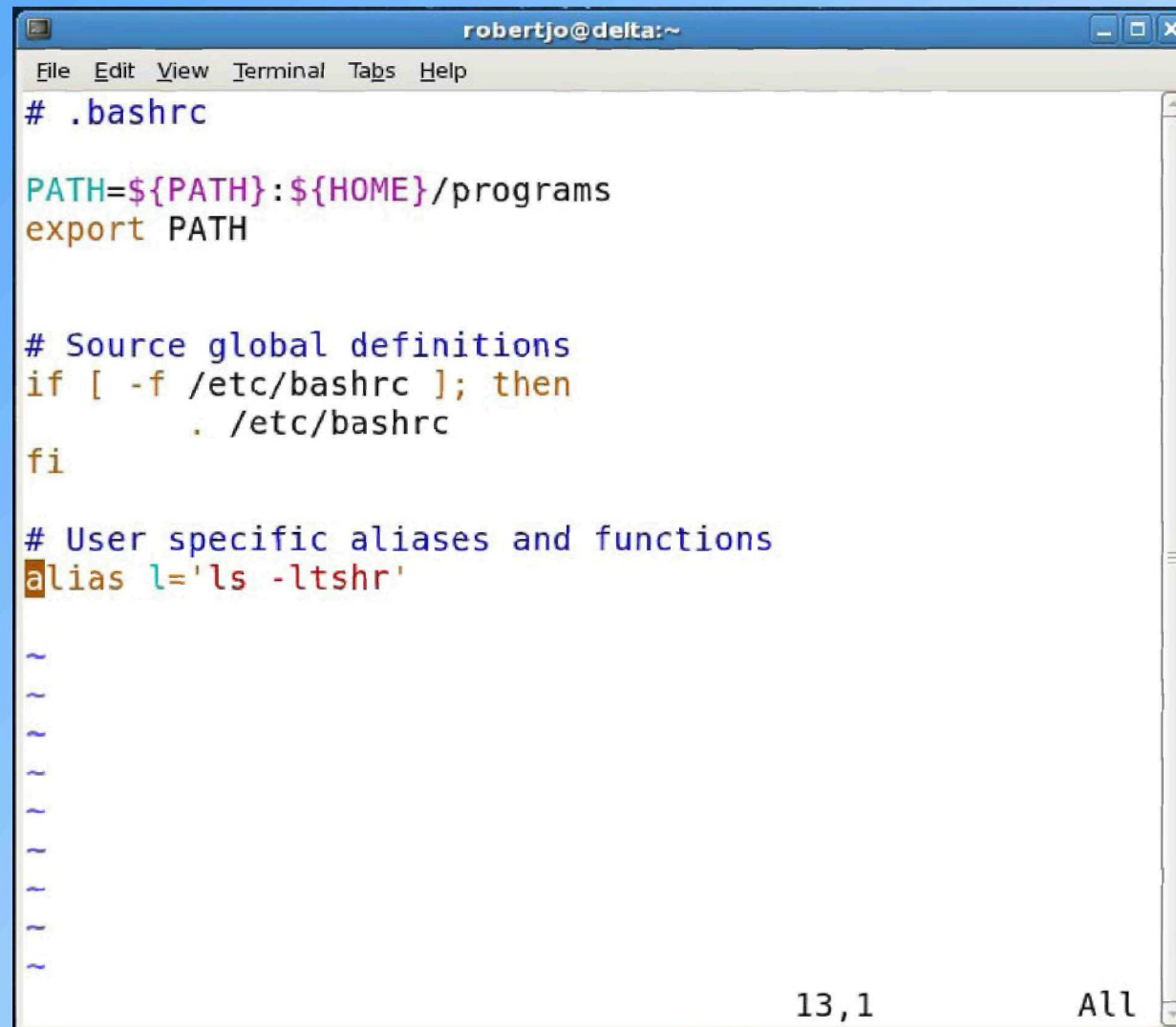
Create an environment variable: `export VAR`

User's home directory: `${HOME}`

Search path for executables: `${PATH}`

.bashrc File

- Behavior of shell environment defined in .bashrc file located in \${HOME}

A terminal window titled 'robertjo@delta:~' showing the contents of the .bashrc file. The window has a menu bar with 'File', 'Edit', 'View', 'Terminal', 'Tabs', and 'Help'. The text in the terminal is as follows:

```
# .bashrc

PATH=${PATH}:${HOME}/programs
export PATH

# Source global definitions
if [ -f /etc/bashrc ]; then
    . /etc/bashrc
fi

# User specific aliases and functions
alias l='ls -ltshr'

~
~
~
~
~
~
~
~
~
~
```

At the bottom right of the terminal window, the text '13,1' and 'All' is visible.

Shell Scripts

- A series of commands incorporated into a text file.
- Executed like a program; evaluated line by line.
- Syntax same as on console.
- Can be used to automate (repetitive) tasks.
 - sh/bash/csh/tcsh/ksh/zsh
 - sed – Replace text (regular expressions)
 - Syntax similar to that used in VI
 - awk – Manipulate column formatted data
 - Syntax similar to C

Conditionals and Loops

- Required for complex or repetitive operations.
- Particularly useful in scripts.
- Branching with “if”:

```
if [ -z "${LD_LIBRARY_PATH}" ] ; then
    LD_LIBRARY_PATH=/opt/mpi/lib
else
    LD_LIBRARY_PATH=${LD_LIBRARY_PATH}:/opt/mpi/lib
fi
```

- Loops:

```
while [ "${var} != "0" ] ; do
    [...]
done
```

```
for var in 1 2 3 4 ; do
    echo ${var}
done
```

Strings and Quoting

- Command line text is split into “words” by whitespace.
- Command line text is split into “sentences” by “;” and “enter”
- “Special” characters need a “\” prefix. (e.g. \\$ is a literal “\$”, a plain “\$” is seen as the first character of a variable; same goes for “ “, “;”, “\”, and so on.
- To build strings one can use single or double quotes (‘ or “)
- Text in single quotes is taken literally (no special characters)
- Text in double quotes allows some special characters: for example variables are expanded
- Text in “backwards” quotes is interpreted as command and then replaced with the output of the command. Examples:

```
cmds=`/bin/lis /bin`
```

```
for i in `seq 1 10` ; do echo “iteration: $i” ; done
```

vi Text Editor

**vi has two modes: 1) command mode
2) text entry mode**

Esc → Command mode 'i' → Text entry mode

vi filename

• **dd**

• **u**

• **/pattern**

• **:%s /pattern1/pattern2/**

• **:w output**

• **:wq**

• **:q!**

Opens filename or create it

Erases one line

Undo last modification

Search for “pattern”

Replace “pattern1” with “pattern2”

Saves the file as “output”

Save file and quit

Quit without saving