System Administration

DAY 2 CONTENT

- String processing
- Piping and Redirection
- Text Editors
- Vi Editor
- Environment Variables
- Quoting
- Command Substitution
- Command Alias
- Command History

String processing

wc command displays the number of characters, words, and lines in a specified file.

```
wc[option] [filename]
```

grep command displays the lines of its input that match a pattern given as an argument

```
grep[options] pattern [filename]
```

String processing

cut command cuts fields or columns of text from standard input or the named file and displays the result to standard output

```
cut option[s] [filename]
```

sort command sorts text data after accepting it from either a file or the output of another command and the sorted text is sent to the standard output, with the original file remaining unchanged in the process.

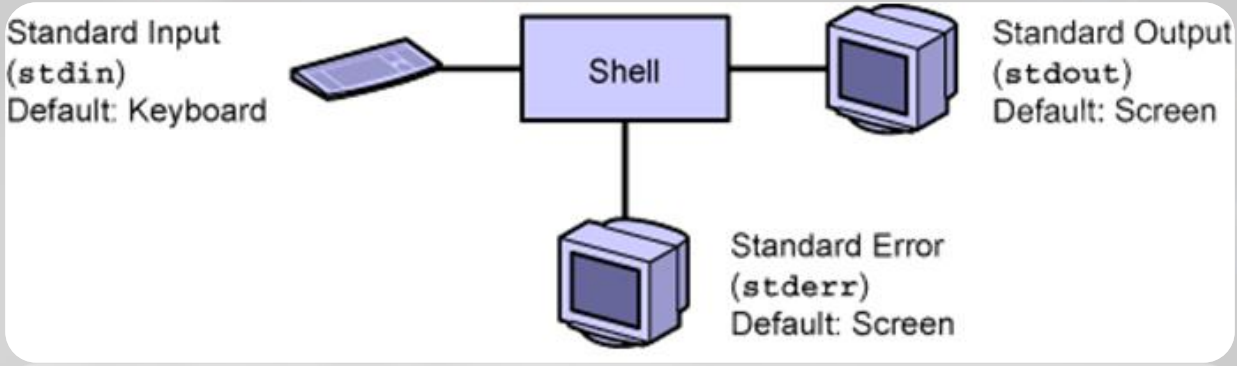
```
sort option[s] [filename]
```

Piping and Redirection

stdin, stdout, and stderr are three data streams created when you launch a Linux command. You can use them to tell if your scripts are being piped or redirected

Redirection metacharacters

- You can redirect input to and output from commands by using redirection.



Piping and Redirection

Redirection symbols

> (1>)
>>
<

Each process that the shell creates, works with **file descriptors**

File descriptors determine where the input to the command originates and where the output and error messages are sent

- 0: stdin
- 1: stdout
- 2: stderr

Piping and Redirection

The Pipe character redirect the standard output to the standard input of another command

```
command | command
```

Example

```
ls -l /usr/bin | more
```

The tee command reads from the standard input and writes to the standard output and a file

Example

```
ls -lR / | tee fname | more
```

Quiz

how can I save the sorted output into a file with two ways?

File Editors

The **gedit** text editor is a graphical tool for editing text files.

Vi editor is used when the desktop environment window system is not available.

Default editor in all UNIX operating systems.

Usually the only editor available in emergencies.

vi editor

VI has three basic modes:

Command mode

- Default mode
- Perform commands to delete, copy, …

Insert mode

- Enter text into the file
- To access it you have many ways, easiest using (i)

Last line mode

- Advanced editing commands
- To access it, enter a colon (:) while in the command mode

vi editor

Most common commands:

i	Inserts text before the cursor
dd	Cuts the line containing the cursor.
yy	Yank a copy of a line.
p	Put yanked text under the line containing the cursor.
D	Deletes the line from the cursor to the right end of the line.
n,nd	Deletes Lines n through n

vi editor

Most common commands:

/string	Searches forward for the string.
n	Searches for the next occurrence of the string.
N	Searches for the previous occurrence of the string.
:set nu, :set nonu	show and hide line numbers
:w	save the file
:w	new_filesave as new file
:wq, :x, ZZ	save and quit
:q!	quit without saving

Environment variables

The shell assumes whatever follows the dollar sign (**$**) in the command line is a variable and substitutes its value

```
echo $HOME
    /home/naggar
```

To view the contents of all variables by running the **env** command

Environment variables

$PWD	The user current working directory
$SHELL	Path name of the login shell
$USER	Currently logged in user
$HOSTNAME	Name of the computer

$HOME Complete path of the user home directory

```
mkdir $HOME/file1
```

$PATH A colon-separated list of directories used by the shell to look for executable program names

Quoting

Quoting is a process that instructs the shell to mask, or **ignore**, the special meaning of metacharacters.

‘ ’	Strong quoting instruct the shell to ignore all metacharacters.
“ ”	Weak quoting instruct the shell to ignore all metacharacters except **$ ` **
\	A backslash character prevents the shell from interpreting the next character as a metacharacter.

Command Substitution

There are different ways to use command inside a command by using command substitution

```
command [options] “[arguments] $(other command)”
```

```
command [options] “[arguments] `other command`”
```

Command alias

alias command is used to see all set aliases

Creating aliases

```
alias shortcut='command'
```

Remove aliases

```
unalias shortcut
```

Bypass aliases

```
\shortcut
```

Command History

bash stores a history of commands you have entered so that you can recall them later.

The history is stored in the user's home directory and is called **.bash_history** by default.

You can recall commands by pressing the up arrow key

!!	Repeats the last command.
!string	Repeats the last command that started with string.
!n	Repeats a command by its number in history output.
!-n	Repeats a command entered n commands back.

Questions?!

Thank YOU!

boost your career