Basic Shell Commands

Operating System Practice

João Vicente Ferreira Lima

Universidade Federal de Santa Maria
jvlima@inf.ufsm.br
http://www.inf.ufsm.br/~jvlima

March 18, 2019

Outline

Outline

VI IMproved

`vim` is a text editor compatible with `vi`.

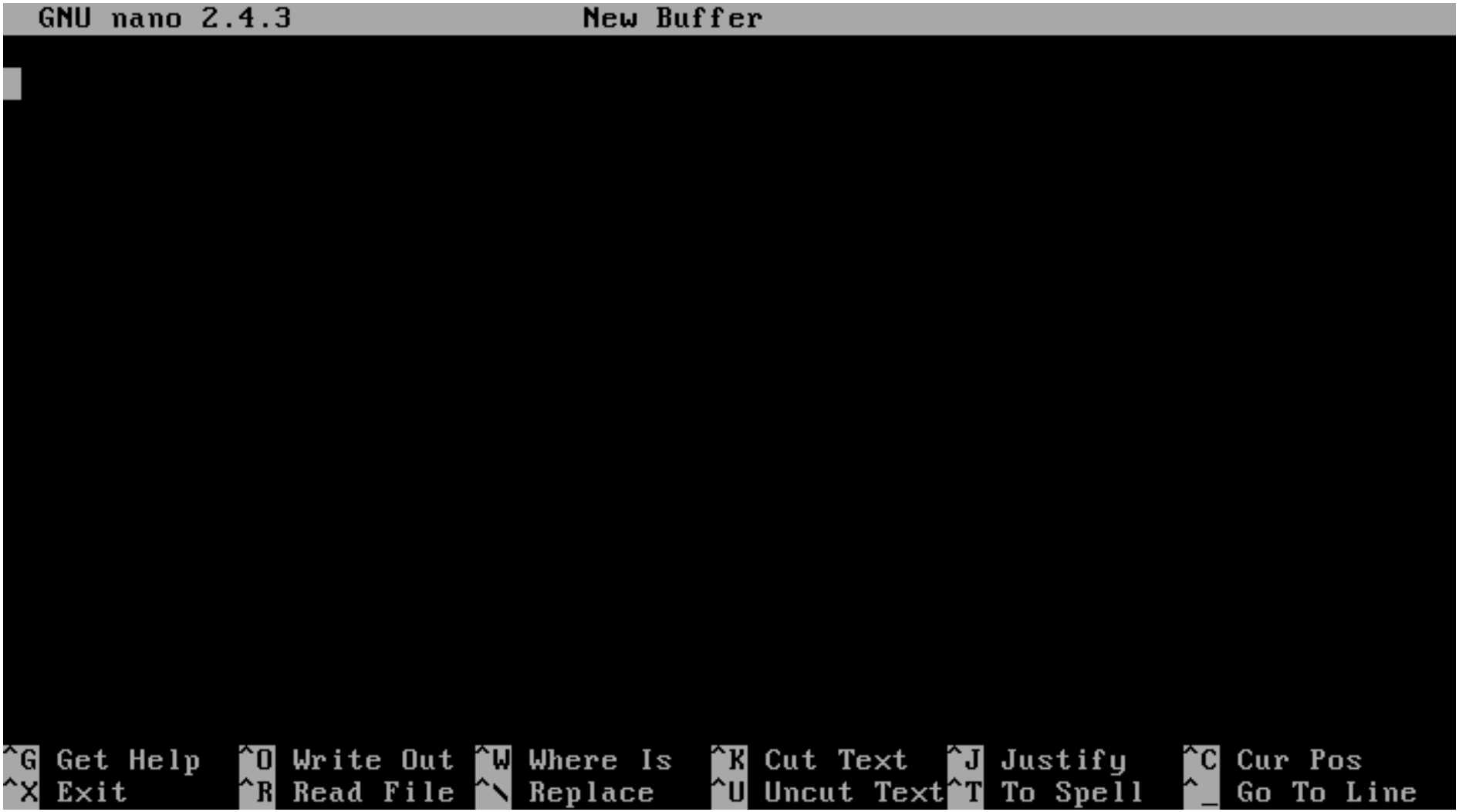
```
vim foo.txt
```

Basic commands:

- `:q` quit
- `:w` write to the current file
- `:w foobar.txt` write to *foobar.txt*
- `:q!` do not save changes
- `:wq` save and quit

nano

`nano` is a small text editor. Its basic commands appears at the screen.



Outline

Help

`man` is the system manual pages.

```
man ls
```

`man -k` search a manual page by keyword.

cat

One of the easiest Linux commands, it simply outputs the contents of one or more files.

```
cat /etc/services
```

ls

The `ls` command lists the contents of a directory.

```
ls
```

```
1_introduction.org
1_introduction.org~
1_introduction.pdf
1_introduction.tex
1_introduction.tex~
2_basic_commands.org
2_basic_commands.org~
2_basic_commands.pdf
2_basic_commands.tex
2_basic_commands.tex~
figures
talk_16_06_21_Inria_journees_scientifiques.org
```

ls

Command options:

- `ls -l` detailed (long) listing
- `ls -color=auto` colored output (GNU only)
- `ls -F` file type information
- `ls -s` print allocated size in blocks
- `ls -h` human readable sizes with `-s / -l`
- `ls -m` print with comma
- `ls -a` list all entries

cp

`cp` copies files. To copy a file to your home directory:

```
cp foo bar
```

To copy a number of files to you home:

```
cp *.txt /tmp
```

Options:

- `cp -r` copy directories
- `cp -a` archive mode (copy with permissions)
- `cp -i` prompt before overwrite an existing file
- `cp -n` do not overwrite an existing file
- `cp -v` verbose mode

mv

The `mv` (move) command is like cp, but renames a file.

```
mv foo bar
```

Options:

- `mv -v` verbose mode
- `mv -i` prompt before overwrite an existing file

touch

The `touch` command sets the modification and access times of files. If the file does not exist, it is created with default permissions.

```
touch foobar
```

rm

The `rm` command attempts to remove a file.

```
rm foobar
```

Options:

- `rm -v` verbose mode
- `rm -f` remove without confirmation
- `rm -i` request confirmation
- `rm -r` remove directories

echo

The `echo` command prints its arguments to the standard output.

```
echo Hello Hello
```

Directories

`cd` changes the shell's current working directory. If you omit `foodir`, the shell returns to the *home directory*.

```
cd foodir
```

Directories

`cd` changes the shell's current working directory. If you omit `foodir`, the shell returns to the *home directory*.

```
cd foodir
```

`mkdir` creates a new directory.

```
mkdir newdir
```

Directories

`cd` changes the shell's current working directory. If you omit `foodir`, the shell returns to the *home directory*.

```
cd foodir
```

`mkdir` creates a new directory.

```
mkdir newdir
```

`rmdir` removes a directory.

```
rmdir foobar
```

Outline

grep

`grep` command prints the lines from a file or input stream that match an expression.

```
grep root /etc/passwd
```

Options:

- `grep -v` invert matching
- `grep -w` whole words
- `grep -n` print line number
- `egrep` uses a pattern (avoid conflicts with -)
- `zgrep` compressed files

less

`less` shows the contents of a file one screenful at a time. Spacebar goes forward, and q quits.

```
grep ie /usr/share/dict/words | less
```

find and locate

`find` walks a file hierarchy.

```
find /usr -name *.h
```

`locate` searches an index that the system builds periodically.

head and tail

To quickly view a portion of a file, use `head` and `tail`.

```
head /etc/passwd
tail /var/log/messages
```

Options:

- `head -n` or `tail -n` show only *n* lines
- `tail +n` print lines starting at line *n*

sort

`sort` sorts text and binary files by lines. Options:

- `sort -n` numeric sort.
- `sort -r` reverse order.
- `sort -k f1,f2` sort by *f1* field, then *f2* field
- `sort -t char` use *char* as a field separator

sort

`sort` sorts text and binary files by lines. Options:

- `sort -n` numeric sort.
- `sort -r` reverse order.
- `sort -k f1,f2` sort by *f1* field, then *f2* field
- `sort -t char` use *char* as a field separator

Sort only by login

```
grep -v '#' /etc/passwd | sort -k1 -t ':'
```

sort

`sort` sorts text and binary files by lines. Options:

- `sort -n` numeric sort.
- `sort -r` reverse order.
- `sort -k f1,f2` sort by *f1* field, then *f2* field
- `sort -t char` use *char* as a field separator

Sort only by login

```
grep -v '#' /etc/passwd | sort -k1 -t ':'
```

Sort by UID user

```
sort -k3 -t ':' -n /etc/passwd
```

Other commands

- `pwd` outputs the name of the current working directory.
- `diff` shows the differences between two text files.
- `file` determine file type

Outline

Bash variables and environment

The shell can store temporary variables, called *shell variables*, containing string values.

Shell variable

```
FOO=blash
```

Bash variables and environment

The shell can store temporary variables, called *shell variables*, containing string values.

Shell variable

```
FOO=blash
```

An *environment variable* is like a shell variable, but not specific to the shell. The main different between environment and shell variables is that the OS passes all your shell's environment variables to programs that the shell runs.

Enviroment variable

```
FOO=blash
export FOO
```

Bash history

`history` prints the last commands issued in the bash. Commands:

- `CTRL-R` reverse search
- `history -c` clear history
- `!n` execute a command at entry *n*

Shell input and output

Output redirection

To send the output of `ls` to a file instead of the terminal:

```
ls > foo.txt
```

If `foo.txt` exists, the shell erases the original file. To append the output:

```
ls >> foo.txt
```

Shell input and output

Output redirection

To send the output of `ls` to a file instead of the terminal:

```
ls > foo.txt
```

If `foo.txt` exists, the shell erases the original file. To append the output:

```
ls >> foo.txt
```

Pipes

We can also send the output of a command to the input of another command trough *pipes*:

```
head /etc/services | tr a-z A-Z
```

Shell input and output

Output redirection

To send the output of `ls` to a file instead of the terminal:

```
ls > foo.txt
```

If `foo.txt` exists, the shell erases the original file. To append the output:

```
ls >> foo.txt
```

Pipes

We can also send the output of a command to the input of another command trough *pipes*:

```
head /etc/services | tr a-z A-Z
```

Input redirection

It is also possible the *input redirection*:

```
head < /etc/services
```

Outline

Processes

`ps` displays information about all processes in the system. Options:

- `ps x` all your running processes
- `ps ax` all processes on the system
- `ps u` detailed information
- `ps w` show full command names

Processes

`ps` displays information about all processes in the system. Options:

- `ps x` all your running processes
- `ps ax` all processes on the system
- `ps u` detailed information
- `ps w` show full command names

`kill` sends a signal to a process. By default, it sends a `TERM`, or terminate, signal. Options:

- `kill pid` send a `TERM` signal
- `kill -STOP pid` to stop a process
- `kill -CONT pid` to continue a process

poweroff and shutdown

These commands close down the system at a given time.

- `shutdown -h now` system is halted now
- `shutdown -r now` reboot now, similar to `reboot`
- `shutdown -r +30 "System will reboot"` reboot the system in 30 minutes and display a warning message to all users
- `poweroff` equivalent to `shutdown -p now`