Fundamentals of Computing: Lecture 38

Piyush P Kurur
Office no: 224
Dept. of Comp. Sci. and Engg.
IIT Kanpur

November 11, 2009

Summary of last class

- Simple shell program.

```
$ cmd arg1 arg2 arg3
```

Summary of last class

- Simple shell program.

  ```
  $ cmd arg1 arg2 arg3
  ```

- `stdin`/`stdout`/`stderr` redirection

Summary of last class

- Simple shell program.

```
$ cmd arg1 arg2 arg3
```

- `stdin/stdout/stderr` redirection

```
$ cmd arg1 arg2 <foo   # get input from foo
$ cmd arg >bar
```

Summary of last class

- Simple shell program.

```
$ cmd arg1 arg2 arg3
```

- stdin/stdout/stderr redirection

```
$ cmd arg1 arg2 <foo   # get input from foo
$ cmd arg >bar

$ cmd 2>&1 | less      # 0-stdin 1-stdout 2-stderr
$ cmd >>foo            # append to the file foo instea
```

Summary of last class

- Simple shell program.

```
$ cmd arg1 arg2 arg3
```

- stdin/stdout/stderr redirection

```
$ cmd arg1 arg2 <foo    # get input from foo
$ cmd arg >bar

$ cmd 2>&1 | less       # 0-stdin 1-stdout 2-stderr
$ cmd >>foo             # append to the file foo insteа
```

- Pipes

```
$ cmd1 | cmd2 | cmd3 | cmd
```

Shell variables

To assign a shell variable

```
$ foo=bar     # no space between foo and bar
```

Shell variables

To assign a shell variable

```
$ foo=bar    # no space between foo and bar
```

To get the value of foo use `$foo`

```
$ echo the value of variable foo is $foo
```

Shell variables

To assign a shell variable

```
$ foo=bar    # no space between foo and bar
```

To get the value of foo use `$foo`

```
$ echo the value of variable foo is $foo
```

Some important shell variables

- `PATH` The directories where an executable is searched

  ```
  $ export PATH=/bin/:/usr/bin:/usr/local/bin:
  ```

- `PS1` The first prompt
- `PS2` the second prompt etc

Funny characters

Shell interpretes some characters differently We have already seen some <, >, &, | etc.

Funny characters

Shell interpretes some characters differently We have already seen some <, >, &, | etc. The character '*' means any sequece of characters

Funny characters

Shell interpretes some characters differently We have already seen some <, >, &, | etc. The character '*' means any sequece of characters

```
$ ls *.c
$ rm *.o
```

Protecting characters

- Use \ to protect funny characters

Protecting characters

- Use \ to protect funny characters

```
$ ls filename\ with\ spaces
$ echo I am an invisible file > \
```

Protecting characters

- Use \ to protect funny characters

```
$ ls filename\ with\ spaces
$ echo I am an invisible file > \
```

- Quoting with '

Protecting characters

- Use \ to protect funny characters

```
$ ls filename\ with\ spaces
$ echo I am an invisible file > \ 
```

- Quoting with '

```
$ ls 'filename with spaces'
$ rm '*'   # actually removes a file called *
$ echo I am an invisible file > ' '
```

Protecting characters

- Use \ to protect funny characters

```
$ ls filename\ with\ spaces
$ echo I am an invisible file > \
```

- Quoting with '

```
$ ls 'filename with spaces'
$ rm '*'   # actually removes a file called *
$ echo I am an invisible file > ' '
```

- Double quoting ". Similar to ' but shell variables expand

```
$ foo=bar
$ echo '$foo is the value of foo'
$ ech "$foo is the value of foo"
```

The program `grep`

grep is a *filter*. It sends only those lines where a given pattern matches.

The program grep

grep is a *filter*. It sends only those lines where a given pattern matches.

```
    grep PATTERN file1 file2 file3
    grep PATTERN
```

The pattern can be what is called a regular expression

The program grep

grep is a *filter*. It sends only those lines where a given pattern matches.

```
grep PATTERN file1 file2 file3
grep PATTERN
```

The pattern can be what is called a regular expression eg. Print all the hidden files (files with name staring with a .)

The program grep

grep is a *filter*. It sends only those lines where a given pattern matches.

```
    grep PATTERN file1 file2 file3
    grep PATTERN
```

The pattern can be what is called a regular expression eg. Print all the hidden files (files with name staring with a .)

```
ls | grep '^\..*' | less #
```

The program grep

grep is a *filter*. It sends only those lines where a given pattern matches.

```
    grep PATTERN file1 file2 file3
    grep PATTERN
```

The pattern can be what is called a regular expression eg. Print all the hidden files (files with name staring with a .)

```
ls | grep '^\..*' | less #
```

- `^` means start of the line
- `.` means any character
- `r *` means many `r`'s
- I have written the `\.` to *escape* the special meaning