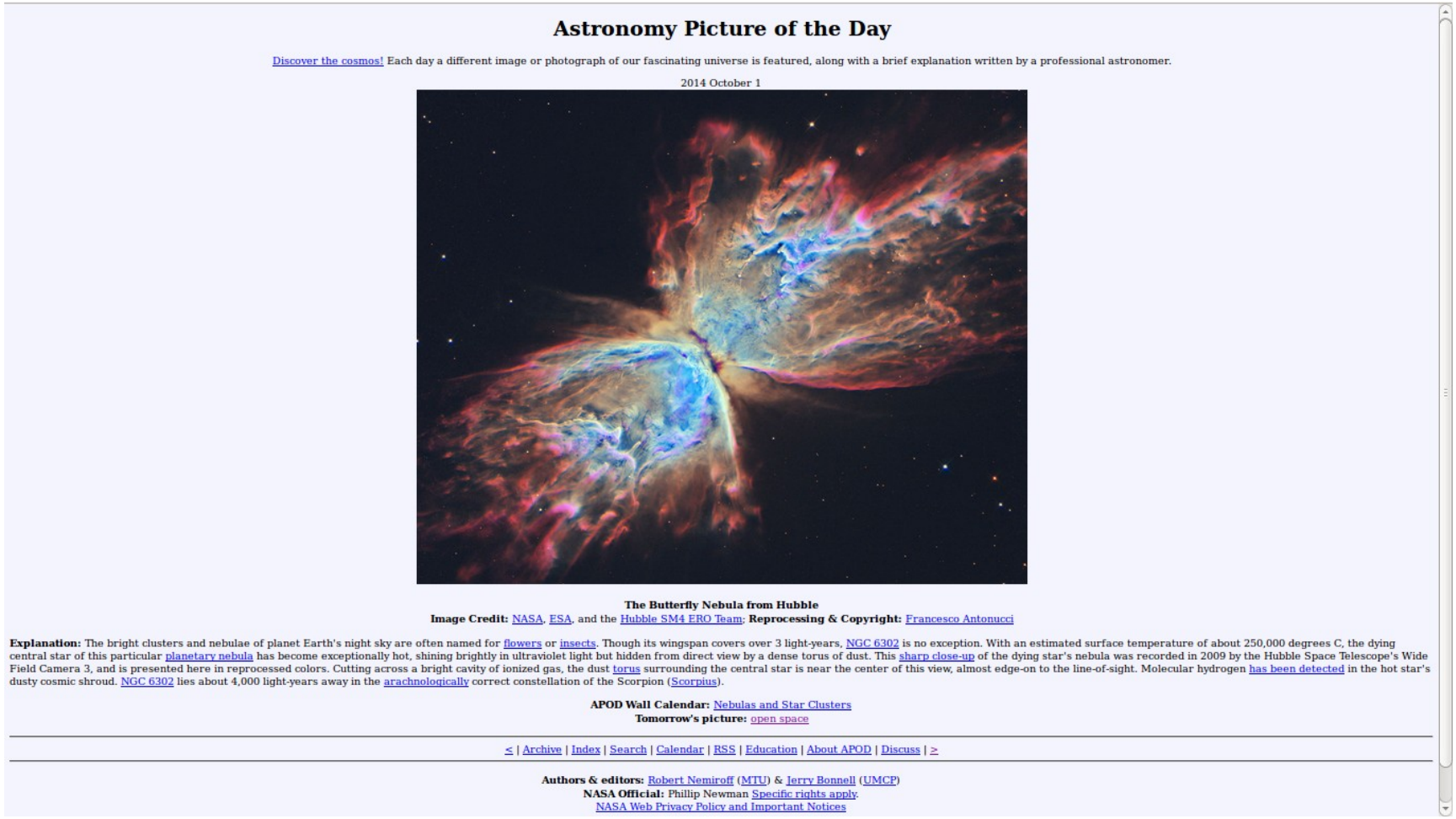
Regular Expressions Practical Example: Where does a page link to?

- Download the apod example (apod.html)



Astronomy Picture of the Day

Discover the cosmos! Each day a different image or photograph of our fascinating universe is featured, along with a brief explanation written by a professional astronomer.

2014 October 1

The Butterfly Nebula from Hubble

Image Credit: NASA, ESA, and the Hubble SM4 ERO Team; **Reprocessing & Copyright:** Francesco Antonucci

Explanation: The bright clusters and nebulae of planet Earth's night sky are often named for flowers or insects. Though its wingspan covers over 3 light-years, NGC 6302 is no exception. With an estimated surface temperature of about 250,000 degrees C, the dying central star of this particular planetary nebula has become exceptionally hot, shining brightly in ultraviolet light but hidden from direct view by a dense torus of dust. This sharp close-up of the dying star's nebula was recorded in 2009 by the Hubble Space Telescope's Wide Field Camera 3, and is presented here in reprocessed colors. Cutting across a bright cavity of ionized gas, the dust torus surrounding the central star is near the center of this view, almost edge-on to the line-of-sight. Molecular hydrogen has been detected in the hot star's dusty cosmic shroud. NGC 6302 lies about 4,000 light-years away in the arachnologically correct constellation of the Scorpion (Scorpius).

APOD Wall Calendar: Nebulas and Star Clusters
Tomorrow's picture: open space

< | Archive | Index | Search | Calendar | RSS | Education | About APOD | Discuss | >

Authors & editors: Robert Nemiroff (MTU) & Jerry Bonnell (UMCP)
NASA Official: Phillip Newman Specific rights apply.
NASA Web Privacy Policy and Important Notices

Regular Expressions: Practical Example: Where does a page link to?

- Write a regular expression that looks for the <a href=”link”> tags in the webpage's HTML.

```
47
48 <b> Explanation: </b>
49 The bright clusters and nebulae of planet Earth's night sky are often
50 named for <a href="http://apod.nasa.gov/apod/ap120929.html">flowers</a> or
51 <a href="http://apod.nasa.gov/apod/ap100425.html">insects</a>.
52
53 Though its wingspan covers over 3 light-years,
54 <a href="http://en.wikipedia.org/wiki/NGC_6302">NGC 6302</a> is no
55 exception.
56
57 With an estimated surface temperature of about 250,000 degrees C,
58 the dying central star of this particular
59 <a href="http://apod.nasa.gov/apod/ap040207.html">planetary nebula</a> has become exceptionally
60 hot, shining brightly in ultraviolet light but hidden from
61 direct view by a dense torus of dust.
62
63 This
64 <a href="http://internal.hubblesite.org/newscenter/archive/releases/2009/25/image/f/">sharp close-up</a>
65 of the dying star's nebula was recorded
66 in 2009 by the Hubble Space Telescope's Wide Field Camera 3,
67 and is presented here in reprocessed colors.
68
69 Cutting across a bright cavity of ionized gas, the dust
70 <a href="http://www.math.tamu.edu/%7Etkiffe/calc3/revolution3/revolution3.html">torus</a>
71 surrounding the central star is near
72 the center of this view, almost edge-on to the line-of-sight.
```

Regular Expressions: Practical Example: Where does a page link to?

- Write a regular expression that looks for the `<a href=”link”>` tags in the webpage's HTML.
- **Hint** - Open the file with:
  - `content = open('apod.html','r').read()`
    - In future could read it from the website:

```
import urllib
url = 'http://apod.nasa.gov/apod/ap141001.html'
webpage = urllib.urlopen(url)
html = webpage.read()
```

Regular Expressions: Practical Example: Where does a page link to?

- Write a regular expression that looks for the `<a href=”link”>` tags in the webpage's HTML.
- Open the file with:
  - `content = open('apod.html','r').read()`
- **Hint** – `re.findall` with a subexpression will return a list of all the examples of that subexpression in the input string.

Regular Expressions: Practical Example: Where does a page link to?

- Write a regular expression that looks for the `<a href=”link”>` tags in the webpage's HTML.

```
content = open('apod.html','r').read()
a = re.findall('<a href="(.*)">',content)
print "\n".join(a)
```

```
...
http://hubblesite.org/servicing_mission_4/
mailto:%20Francesco%20dot%20antonucci%20at%20fastwebnet%20dot%20it
http://apod.nasa.gov/apod/ap120929.html
...
http://www.phy.mtu.edu/
http://antwrp.gsfc.nasa.gov/htmltest/jbonnell/www/bonnell.html">Jerry Bonnell</a>
(<a href="http://www.astro.umd.edu/
http://apod.nasa.gov/apod/lib/about_apod.html#srapply
..
```

Regular Expressions: Practical Example: Where does a page link to?

- Write a regular expression that looks for the `<a href=”link”>` tags in the webpage's HTML.

```
content = open('apod.html','r').read()
a = re.findall('<a href="(.*)">',content)
print "\n".join(a)
```

- Fix regular expression...

```
'<a href="(.*)">'
```

Regular Expressions: Practical Example: Where does a page link to?

- Write a regular expression that looks for the `<a href=”link”>` tags in the webpage's HTML.

```
content = open('apod.html','r').read()
a = re.findall('<a href="(.*)">',content)
print "\n".join(a)
```

- Fix regular expression...

```
'<a href="([^\'"]*)">'
```

Regular Expressions: Practical Example: Where does a page link to?

- Write a regular expression that looks for the `<a href=”link”>` tags in the webpage's HTML.

```
content = open('apod.html','r').read()
a = re.findall('<a href="(.*)">',content)
print "\n".join(a)
```

- Fix regular expression...

```
'<a href="(http[^\'"]*)">'
```

Regular Expressions: Practical Example: Where does a page link to?

- Write a regular expression that looks for the `<a href=”link”>` tags in the webpage's HTML.

```
content = open('apod.html','r').read()
a = re.findall('<a href="(.*)">',content)
print "\n".join(a)
```

- Fix regular expression...

```
'<a href="(http[^\'"]*)">'
```

  - Could also match both " and ' using `["\']`?

Regular Expression: regex golf puzzle 1

MATCH

- buffoon
- foody
- foolish
- fool
- catfoot
- afoot
- footlights
- footmen
- seafood

REJECT

- crinkle
- palace
- wildfowl
- critical
- spontaneous
- forget
- info

Regular Expression: regex golf puzzle 1

MATCH

- buffoon
- foody
- foolish
- fool
- catfoot
- afoot
- footlights
- footmen
- seafood

REJECT

- crinkle
- palace
- wildfowl
- critical
- spontaneous
- forget
- info

Regular Expression: regex golf puzzle 2

MATCH

- mick
- rick
- trick
- candlestick
- crick

REJECT

- strategic
- tricked
- picked
- icky
- yickicky

Regular Expression: regex golf puzzle 2

MATCH

- mick
- rick
- trick
- candlestick
- crick

REJECT

- strategic
- tricked
- picked
- icky
- yickicky

`ick$`

Regular Expression: regex golf puzzle 3

MATCH

- abac
- accede
- adead
- babe
- bead
- bebed
- bedad
- bedded
- bedead

letters a-f occur 4 times or more in a row.

REJECT

- beam
- buoy
- canjac
- chymia
- corah
- cupula
- griece
- hafter
- idic

Regular Expression: regex golf puzzle 3

MATCH

- abac
- accede
- adead
- babe
- bead
- bebed
- bedad
- bedded
- bedead

letters a-f occur 4 times or more in a row.

REJECT

- beam
- buoy
- canjac
- chymia
- corah
- cupula
- griece
- hafter
- idic

```
[a-f]{4}
```

Regular Expression: regex golf puzzle 4

MATCH

- l**abab**ab
- r**abab**
- n**abab**ababa
- la**abab**i
- **abab**baba

abab occurs in these.

REJECT

- abba
- rambam
- abraham
- abridge
- albright
- cab
- bracket

Regular Expression: regex golf puzzle 4

MATCH

- l**abab**ab
- r**abab**
- n**abab**ababa
- la**abab**i
- **abab**baba

abab occurs in these.

REJECT

- abba
- rambam
- abraham
- abridge
- albright
- cab
- bracket

`abab` or `(ab){2}`

Regular Expression: regex golf puzzle 5

MATCH

- `wires`
- `these`
- `words`
- `maybe`
- `fives`
- `count`
- `doggy`
- `grump`
- `trips`
- `tight`

match words have five.

REJECT

- the
- word
- you
- need
- to
- reject
- doesn't
- have
- five
- letters
- in
- it

Regular Expression: regex golf puzzle 5

MATCH

- `wires`
- `these`
- `words`
- `maybe`
- `fives`
- `count`
- `doggy`
- `grump`
- `trips`
- `tight`

match words have five.

REJECT

- the
- word
- you
- need
- to
- reject
- doesn't
- have
- five
- letters
- in
- it

```
^.{5}$
```

Regular Expression: regex golf puzzle 7

MATCH

- thingandthing
- stuffandstuff
- blahandblah
- abcabc
- tootleandpootle
- rappingandtapping
- beepbeep

some text, then maybe sometihng, then the same text again.

REJECT

- granulation
- secret
- solution
- bumbling
- transient
- rectilinear
- convolve

Regular Expression: regex golf puzzle 7

MATCH

- **thing**and**thing**
- **stuff**and**stuff**
- **blah**and**blah**
- **abcabc**
- t**ootle**andp**ootle**
- r**apping**andt**apping**
- **beepbeep**

some text, then maybe sometihng, then the same text again.

REJECT

- granulation
- secret
- solution
- bumbling
- transient
- rectilinear
- convolve

```
(.{3,}).*\1
```

Regular Expression: regex golf puzzle 7

MATCH

- **thing**and**thing**
- **stuff**and**stuff**
- **blah**and**blah**
- **abcabc**
- t**ootle**andp**ootle**
- r**apping**andt**apping**
- **beepbeep**

some text, then maybe sometihng, then the same text again.

REJECT

- granulation
- secret
- solution
- bumbling
- transient
- rectilinear
- convolve

```
(.{3,}).*\1
```

Regular Expression: regex golf puzzle 8 - skip!

Regular Expression: regex golf puzzle 9

MATCH

- a**ll**ochira**ll**y
- an**ti**covenan**ti**ng
- b**ar**b**ar**y
- c**al**electric**al**
- e**nt**ableme**nt**
- **et**han**et**hiol
- fr**ou**fr**ou**
- f**ur**f**ur**yl
- g**al**ag**al**a
- h**ea**vyh**ea**ded
- l**in**guatul**in**e

two letter substring appears twice

REJECT

- granulation
- secret
- solution
- bumbling
- transient
- rectilinear
- convolve

Regular Expression: regex golf puzzle 9

MATCH

- a**ll**ochira**ll**y
- an**ti**covenan**ti**ng
- b**ar**b**ar**y
- c**al**electric**al**
- e**nt**ableme**nt**
- **et**han**et**hiol
- fr**ou**fr**ou**
- f**ur**f**ur**yl
- g**al**ag**al**a
- h**ea**vyh**ea**ded
- l**in**guatul**in**e

two letter substring appears twice

REJECT

- granulation
- secret
- solution
- bumbling
- transient
- rectilinear
- convolve

```
(..).*\1
```

Regular Expression: regex golf puzzle 10

MATCH

- `abba`
- `toot`
- `roor`
- `dood`
- `meem`
- `alla`

Whole string consists of two subexpressions, each of one letter repeated in reverse.

REJECT

- `trait`
- `repper`
- `pepper`
- `llill`
- `abab`
- `abbr`
- `rapr`
- `raari`
- `abbar`

```
^(.)(.)\2\1$
```

The Command Line

The bash command line interface (CLI)

- Tool which tells the computer commands by typing the commands at a command prompt.
- Allows complex commands that wouldn't be possible with a GUI, and allows repetitive tasks to be automated.
- Standard component of linux/unix systems.

- Lots of resources online, an easy start is at:

  http://linuxcommand.org/learning_the_shell.php

The book associated with this is available on muele

The bash command line interface (CLI)

- Far too large a topic to cover in depth.
- Areas to cover:
  - Navigation (cd, pwd, ls)
  - The file system (/ /etc /bin /var /home /tmp /mnt)
  - File manipulation (cp, mv, rm, mkdir)
  - ~~I/O redirection (stdin, stdout, pipes)~~
  - ~~Permissions~~
  - ~~Job Control~~

Beyond scope of course. The TLCL pdf tutorial covers these topics.

The book associated with this is available on muele

Install or connect remotely...

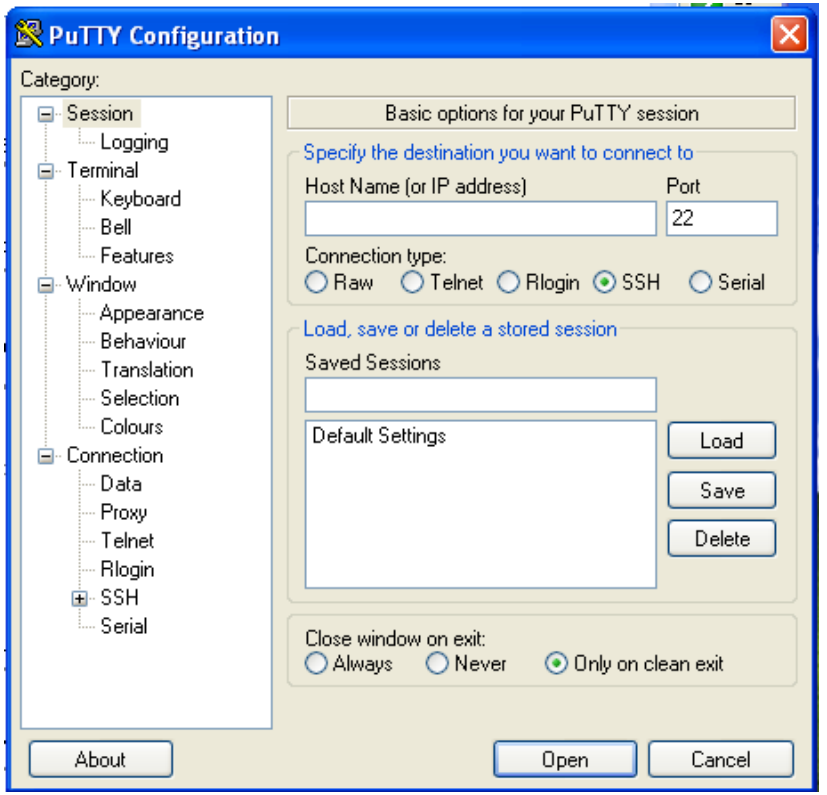
Installation:

- If you're using linux or unix you'll already have it.
- If you're using windows there are several choices:
  - Install **linux** (e.g. ubuntu) dual-boot
  - Install **ubuntu** as a virtual machine (e.g. using virtualBox)
  - Install a toolset like **cygwin**
  - **I've included a slightly out-of-date cygwin-lite.exe**
  - Find a linux server you can access with **ssh**

- During the lecture you can **ssh** into my laptop...

Install or connect remotely...

- During the lecture you can **ssh** into my laptop:
  - `ip address:` **`10.10.?.?`** `port` **`22`**
  - `Username:` **`advprog`**
  - `Password:` **`b4shpr4c`**
  - Download putty to connect to it with.



The CLI: Navigation

- `pwd` - print working directory
- `ls` - list
- `cd` - change directory

- Try them out...

The CLI: Navigation

- `pwd` - print working directory
- `ls` - list
- `cd` - change directory

```
advprog@atlas:~$ pwd
/home/advprog
advprog@atlas:~$ ls
pythonwork  welcome.txt
advprog@atlas:~$ cd pythonwork
advprog@atlas:~/pythonwork$ pwd
/home/advprog/pythonwork
advprog@atlas:~/pythonwork$ ls
test.py
advprog@atlas:~/pythonwork$
```

Changing directory

- In the last example we used a relative path:

  `cd pythonwork`

- We can also use the location's absolute path:

  `cd /home/advprog/pythonwork`

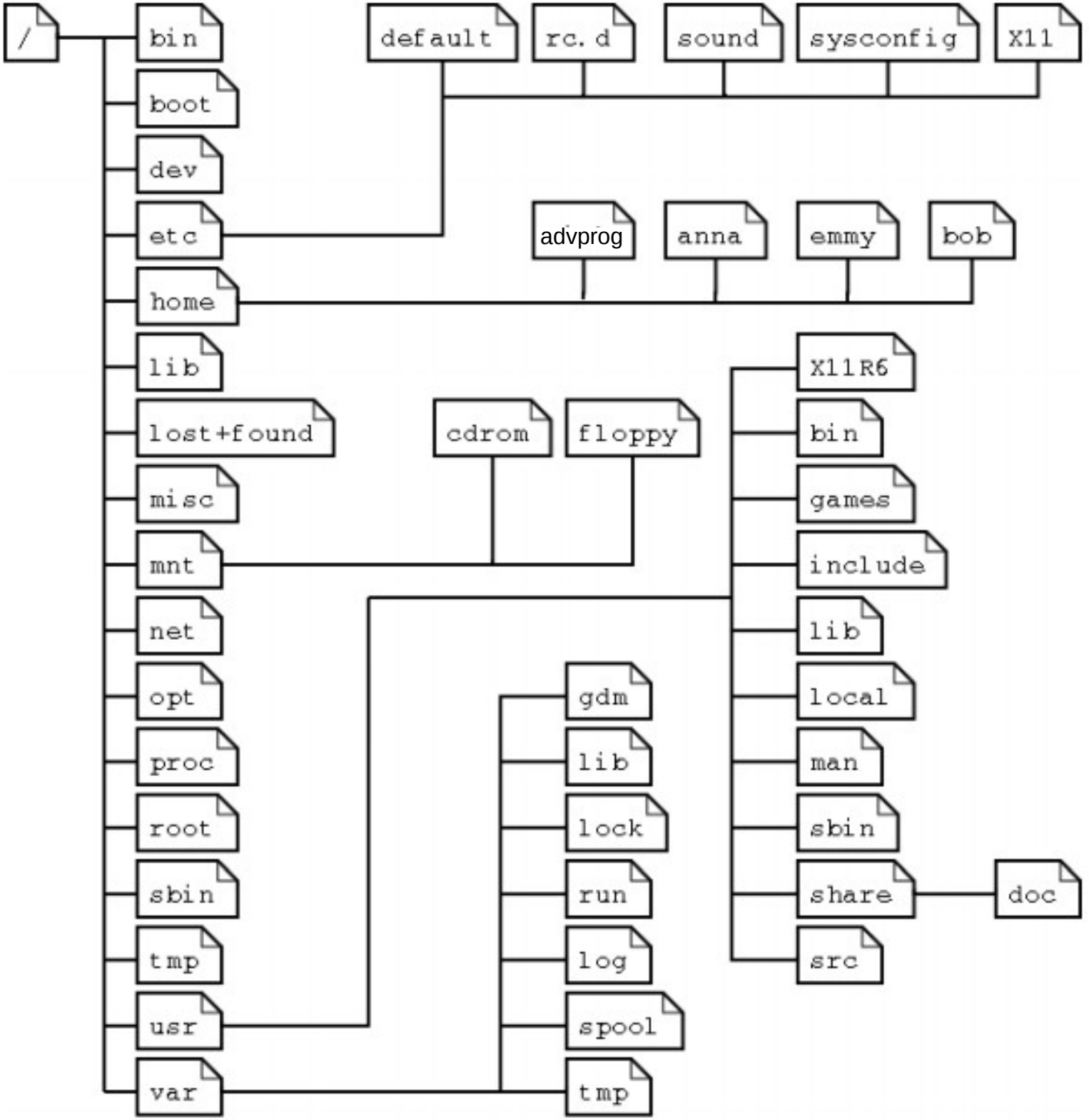
- **`/home/advprog`** is our home directory
  - this can be referred to as **`~`**
- For example, to get back to the home directory, type:

  `cd ~`

  (note: just typing **`cd`** by itself also takes you back to the home directory)

Changing directory

```
advprog@atlas:~/pythonwork$ pwd
/home/advprog/pythonwork
advprog@atlas:~/pythonwork$ cd ~
advprog@atlas:~$ pwd
/home/advprog
advprog@atlas:~$
```


/
bin
boot
dev
etc
home
lib
lost+found
misc
mnt
net
opt
proc
root
sbin
tmp
usr
var
default
rc.d
sound
sysconfig
X11
advprog
anna
emmy
bob
cdrom
floppy
gdm
lib
lock
run
log
spool
tmp
X11R6
bin
games
include
lib
local
man
sbin
share
doc
src

Moving around the directory tree

- Relative paths let you go 'up' the tree, using ..
  - For example `cd ..` moves 'up' one level.
- There is a folder called `bashpractice` in advprog's home directory.
  - To get to it, from where-ever we are, we can use:

    `cd ~/bashpractice`
- Use ls to see what's in there.

```
advprog@atlas:/$ cd ~/bashpractice
advprog@atlas:~/bashpractice$ ls
fiction  nonfiction
advprog@atlas:~/bashpractice$
```

Moving around the directory tree

- Use cd and ls to explore the tree, you can go down into a directory by typing, e.g.

  `cd fiction`

- You can go up a directory by typing `cd ..`
- You can also try going straight up and down by using `cd ../nonfiction`.
- What relative path would get from:

  `~/bashpractice/nonfiction/art`

  to

  `~/bashpractice/fiction/scifi`

Moving around the directory tree

- Use cd and ls to explore the tree, you can go down into a directory by typing, e.g.

  `cd fiction`

- You can go up a directory by typing `cd ..`
- You can also try going straight up and down by using `cd ../nonfiction`.
- What relative path would get from:

  `~/bashpractice/nonfiction/art`

  to

  `~/bashpractice/fiction/scifi`

`cd ../../fiction/scifi`

Moving around the directory tree

- Explore upwards as well, what is above your home directory?

```
cd ..
cd /
ls
```

Use tab-completion to type commands quickly.

Linux is case-sensitive.

Spaces in filenames make things difficult!

Files that start with a . are hidden to see them use

```
ls -a
```

Looking around

- We've already seen we can use ls to list the files in a directory.
  - Try `ls -l` to list them in more detail.

```
advprog@atlas:~$ ls -l
total 12
drwxrwxr-x 4 advprog advprog 4096 Oct  4 19:25 bashpractice
drwxrwxr-x 2 advprog advprog 4096 Oct  4 19:10 pythonwork
-rw-rw-r-- 1 advprog advprog   59 Oct  4 19:08 welcome.txt
advprog@atlas:~$
```

Looking around

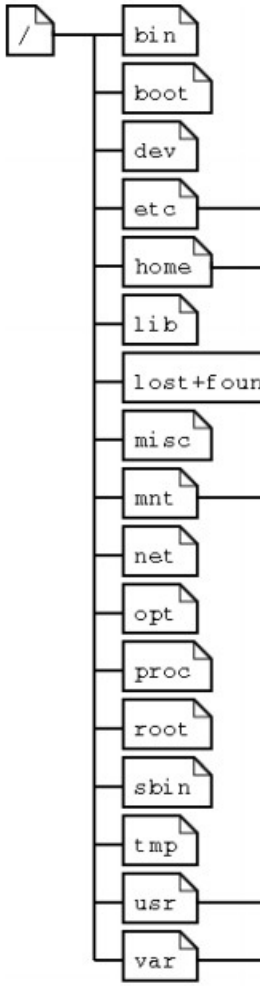
- We've already seen we can use ls to list the files in a directory.
  - Try `ls -l` to list them in more detail.

```
drwxrwxr-x 4 advprog advprog 4096 Oct  4 19:25 bashpractice
drwxrwxr-x 2 advprog advprog 4096 Oct  4 19:10 pythonwork
-rw-rw-r-- 1 advprog advprog   59 Oct  4 19:08 welcome.txt
----------   ------- -------  --- ------------ -------------
permissions   owner   group  file mod-date/time  filename
                             size
```

Looking around

- Look in a text file with **`cat`**, **`less`** or **`nano`**

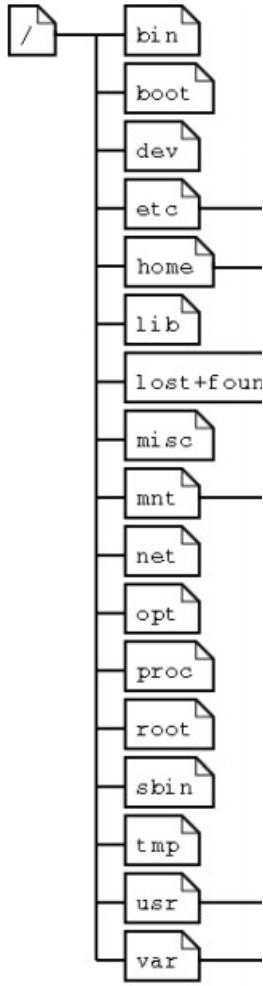
The file system again...



Directory	Comments
`/`	The root directory. Where everything begins.
`/bin`	Contains binaries (programs) that must be present for the system to boot and run.
`/boot`	Contains the Linux kernel, initial RAM disk image (for drivers needed at boot time), and the boot loader. Interesting files: • `/boot/grub/grub.conf` or `menu.lst`, whicl are used to configure the boot loader. • `/boot/vmlinuz`, the Linux kernel
`/dev`	This is a special directory which contains *device nodes*. "Everything is a file" also applies to devices. Here is where the kernel maintains a list of all the devices it understands.

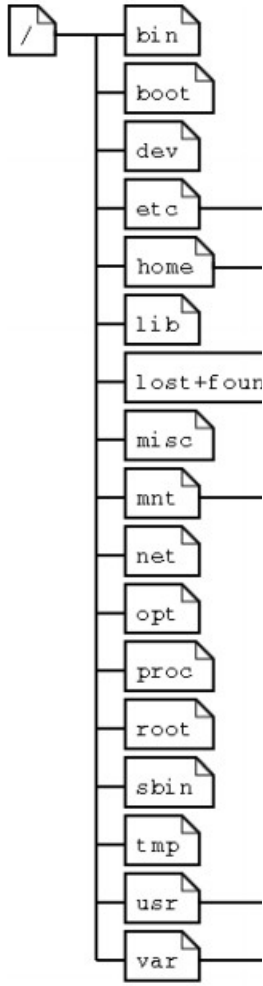
The file system again...

`/etc`	The `/etc` directory contains all of the system-wide configuration files. It also contains a collection of shell scripts which start each of the system services at boot time. Everything in this directory should be readable text. Interesting files: While everything in `/etc` is interesting, here are some of my all-time favorites: • `/etc/crontab`, a file that defines when automated jobs will run. • `/etc/fstab`, a table of storage devices and their associated mount points. • `/etc/passwd`, a list of the user accounts.
`/home`	In normal configurations, each user is given a directory in `/home`. Ordinary users can only write files in their home directories. This limitation protects the system from errant user activity.
`/lib`	Contains shared library files used by the core system programs. These are similar to DLLs in Windows.



The file system again...

`/tmp`	The `/tmp` directory is intended for storage of temporary, transient files created by various programs. Some configurations cause this directory to be emptied each time the system is rebooted.
`/usr`	The `/usr` directory tree is likely the largest one on a Linux system. It contains all the programs and support files used by regular users.
`/var`	With the exception of `/tmp` and `/home`, the directories we have looked at so far remain relatively static, that is, their contents don't change. The `/var` directory tree is where data that is likely to change is stored. Various databases, spool files, user mail, etc. are located here.
`/var/log`	`/var/log` contains *log files*, records of various system activity. These are very important and should be monitored from time to time. The most useful one is `/var/log/messages`. Note that for security reasons on some systems, you must be the superuser to view log files .



Manipulating files and directories

- cd to the `~/lesson` directory
- In this directory make a folder with a folder name of just spaces, for yourself, e.g:
  - `mkdir mike`
- Copy the darwin.txt file from ~/lesson to your new folder, e.g.:
  - `cp ~/lesson/darwin.txt ~/lesson/mike`
  - You can do this using relative paths! So if I'm in the lesson folder, I can copy the file with:

```
cp darwin.txt mike
```

Manipulating files and directories

- **Danger**! Linux will copy the file and overwrite files without asking or warning you!

Manipulating files and directories

- Display the contents of the file in your own folder with cat, less or nano.
- Can use wildcards to copy or move selections of files...
- In the ~/lesson/test folder there are 9 files, each for a different (fictitious) student. How can we copy just the failed ones to our folder?

```
cd ~/lesson/test
cp *fail* ../mike
```

- **Not quite the same as regular expressions though!!!**

Manipulating files and directories

- Other commands of use:
  - `rm a` remove a - **DANGER**: This deletes without asking or checking, and is irreversible (no recycle bin!)
  - `rm -r a` recursively remove a (required when deleting directories)
  - `mkdir a` make directory called a
  - `cp a .` copy a to the current directory.
  - `cp -r a b` copy recursively (allows directories to be copied).
  - `mv a b` move a to b (also used to rename files)

Random commands to try out...

- **`date`** - gives the date and time
- **`which`** `ls` - tells you which file a particular command runs, in this example looking at 'ls'
- **`man`** `grep` - displays the manual page for a command.

- Up until now, the commandline doesn't seem to have provided much use beyond what a graphical interface can achieve. In next week's lecture we'll look at some of its more powerful tools.

Homework

- Submit your assignment (before Friday!)
- Go through the bash things we've learnt, chapters 1-5 of the TLCL pdf.