Unix Files and Directories

Computer Science Department
CS2211a: Software Tools & Systems Programming
Fall 2013
Instructor: Mahmoud R. El-Sakka
Office: MC-419
Email: elsakka@csd.uwo.ca
Phone: 519-661-2111 x86996

Files and Directories

- What is a computer file?
  - a *container* for *ordered* data
    - essentially a sequence of bytes
    - can be *accessed* in its *order* inside the file
  - *persistent* (stays around) and
  - accessible *by name*
- A Unix file is identified by its file name in a directory
  - this name is actually used to
    - resolve the *hard disk name*, the *cylinder number*, the *track number*, the *sector number*, and the *block number*

Files and Directories

- Unix files come in various flavors, such as
  - *Regular normal file*
  - *Directory*
    - a file containing pointers to other files
    - equivalent of a "folder" on a Mac or Windows
  - *Link*
    - a pointer to another file
    - similar to "shortcuts" in Windows, *but better*
  - *Device*
    - An access to a device (e.g., terminal, soundcard, mouse, …) like a file
- Unix files and directories can be seen as a **tree**, where
  - Files are similar to **leaves**
  - Directories are similar to **nodes**

Directories

- Current Working Directory
  - the directory you are looking at right now
  - the shell remembers this for you
- To determine the Current Working Directory, use the command pwd (Print Working Directory)

  obelix[18] > pwd

  Result: print the current working directory

Directories

- Moving through the filesystem
  - Use the "cd" (Change Directory) command to move between directories and change the current directory

    obelix[19] > cd cs2211

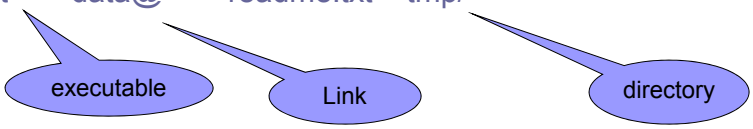
    Result: Makes cs2211 the current working directory
    (of course if cs2211 directory is exist)

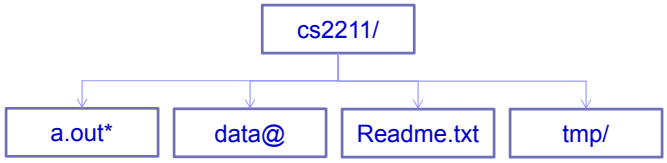
Directories

- Listing the contents of a directory
  - Use the "ls" (LiSt) command to list the contents of a directory

    obelix[20] > ls

    a.out* data@ readme.txt tmp/





Directories

- the Unix filesystem is organized as an *upside-down tree*
  - at the top of the filesystem is the ***root***
    - a lone slash: /
    - this is *not* a backslash (opposite of Windows)!
  - For example, you can change to the root directory:

```
obelix[21] > cd /
obelix[22] > ls
InstallShield/  dev/        home/          obelix/     sbin/
admin@          devices/    kernel/        office/     student/
bin@            etc/        lib/           opt/        system/
boot/           export/     local/         patches/    text@
cata/           faculty/    mnt/           platform/   tmp/
cdrom/          g0usr/      net/           proc/       usr/
csd/            gaul/       noautoshutdown public@     var/
db2users/       guru/       nsr@           s1usr/      vol/
```


Directories

```
obelix[23] > pwd
/
obelix[24] > cd var
obelix[25] > pwd
/var
obelix[26] > ls
adm/      dmi/   ldap/    ntp/       scn/          uucp/
apache/   dt/    lib/     opt/       sma_snmp/     webconsole/
apache2/  fm/    log/     postgres/  snmp/         yp/
audit/    fps/   lp/      preserve/  spool/
cache/    imq/   mysql/   run/       statmon/
cc-ccr/   inet/  news/    sadm/      svc/
crash/    krb5/  nfs/     saf/       tmp/
cron/     ld/    nis/     samba/     tsol/
```

Directories

- Some standard directories and files in a typical Unix system
  - / the root
  - /bin BINaries (executables) actually, it is a symbolic link to /usr/bin
  - /dev DEVices (peripherals)
  - /devices where the DEVICES really live
  - /etc startup and administrative system files
  - /lib LIBraries
  - /opt OPTional software packages
  - /sbin System administration BINaries
  - /tmp place for TeMPorary files
  - /var files that vary as the system runs, e.g., log, spool, mail, …

Directories

- Some standard directories and files in a typical Unix system
  - /usr USeR stuff
  - /usr/bin BINaries again
  - /usr/include include files for compilers
  - /usr/lib LIBraries of functions etc
  - /usr/local local stuff
  - /usr/local/bin local BINaries
  - /usr/local/lib local LIBraries
  - /usr/sbin sysadmin stuff
  - /usr/tmp place for more TeMPorary files
  - /usr/ucb UCB binaries

Pathnames

- A typical Unix file system spans many disks
  - As a user you don't really know, or need to know, which physical disk things are on
    - in fact, you don't even know which machine they are attached to; disks can be "remote" (eg: your home directory is stored on a disk attached to a computer in the machine room somewhere)
    - Look at the df command to see different disks and space used on them
- The absolute path
  - to identify where a file is,
    - start from the root directory, i.e, /
    - string together all directory names that is in your path to the file
    - separating directory names by a single slash each

      e.g. /faculty/elsakka

      this is the absolute path for my home directory
    - Students' home directories are located under /student

Pathnames

- When you first log in, you are in your home directory

```
obelix[1] > pwd
 /faculty/elsakka
```

- Your home directory is also stored in an *environment variable* called HOME

```
obelix[2] > echo My home is $HOME
My home is /faculty/elsakka
```

- You can "*Go Home*" by typing

```
obelix[3] > cd $HOME
```

Pathnames

- Some shorthand
  - In some shells (including tcsh, csh, and bash, *but not* sh), $HOME can be abbreviated as ~ (tilde)
  - Example: obelix[26] > cd ~/bin
    - change to the bin directory under your home directory (equivalent to $HOME/bin)
    - this is where you usually store your own commands or "executables"
  - To quickly go home:

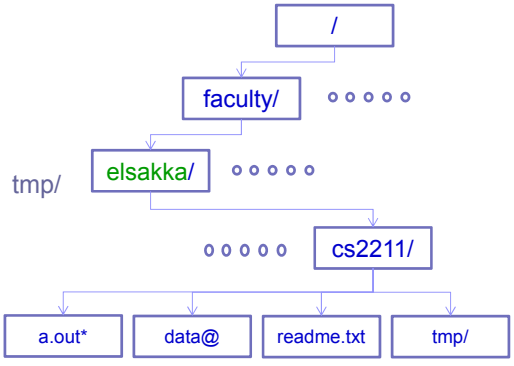
    obelix[27] > cd

    cd with no argument changes the working directory to your home directory
  - ~elsakka refers to the home directory of elsakka
    - ***For me***, ~elsakka is the same as ~, which refers to Mahmoud El-Sakka's home directory (/faculty/elsakka)

Pathnames

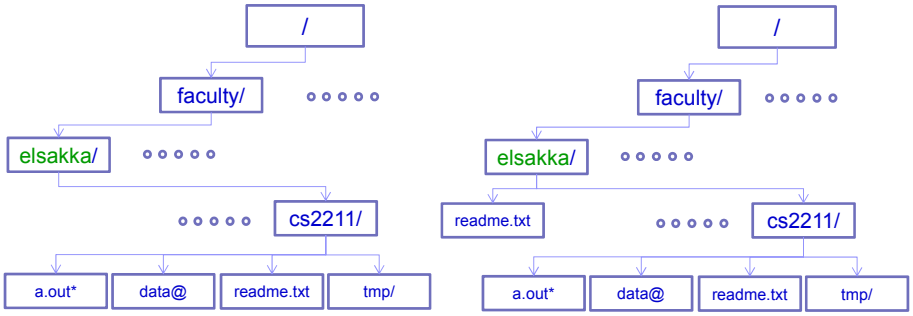
- Relative pathnames
  - You can specify pathnames relative to the current working directory
    - This is called a relative pathname
  - For commands which require a file name, you can specify a pathname (relative or absolute)
  - For example

```
obelix[28] > pwd
/faculty/elsakka
obelix[29] > cd cs2211
obelix[30] > ls
 a.out*  data@     readme.txt    tmp/
obelix[31] > cd tmp
obelix[32] > pwd
/faculty/elsakka/cs2211/tmp
```



Pathnames

- Every directory contains two "*special*" directories: **.** and **..**
  - **.** : another name for the current directory
    - Assume that the current working directory is /faculty/elsakka
      cp cs2211/readme.txt .

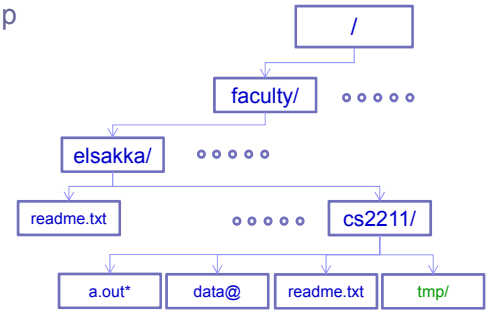


Pathnames

.. : another name for the immediate parent directory of the current directory

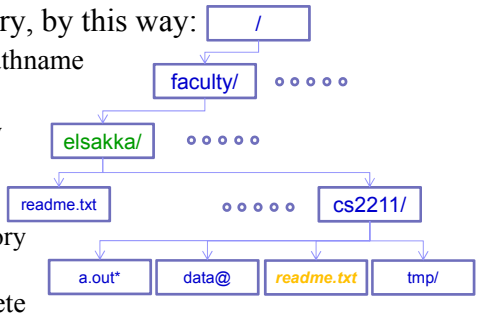
- use this to cd to your parent:

```
obelix[33] > pwd
 /faculty/elsakka/cs2211/tmp
obelix[34] > cd ..
obelix[35] > pwd
 /faculty/elsakka/cs2211
obelix[36] > cd ../..
obelix[37] > pwd
 /faculty
```



Pathnames

- You can locate a file, or a directory, by this way:
  - look at the first character of the pathname
    - / start from the root
    - . start from the current directory
    - .. start from the parent directory
    - ~ start from a home directory
    - O.w. start from the current directory
  - go down to the subdirectories in the pathname, until you complete the whole pathname
  - if you start in ~elsakka, the following are equivalent:
    - /faculty/elsakka/cs2211/readme.txt
    - ~elsakka/cs2211/readme.txt
    - ~/cs2211/readme.txt
    - cs2211/readme.txt
    - ../elsakka/cs2211/readme.txt
    - ../../faculty/elsakka/cs2211/readme.txt



Working with Directories

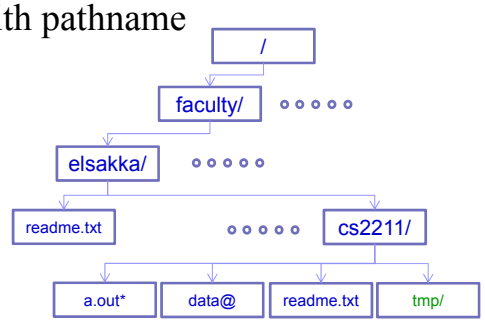
- Create a directory with the mkdir command

  mkdir newdirname

- newdirname can be given with pathname

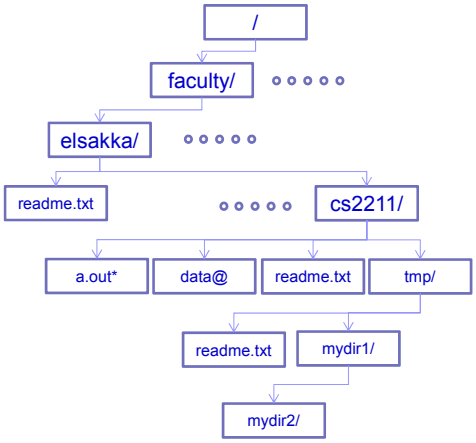
```
obelix[38] > cd ~/cs2211/tmp
obelix[39] > cp ../readme.txt .
obelix[40] > mkdir mydir1
obelix[41] > ls
mydir1/     readme.txt
obelix[42] > mkdir mydir1/mydir2
obelix[43] > ls mydir1
mydir2/
```

we can specify a directory with ls



Before

Working with Directories



After

Working with Directories

- Remove a directory with the rmdir command

  rmdir dirname

  - dirname is the directory to remove and can be specified using a pathname
  - if the directory exists and is ***empty*** it will be removed

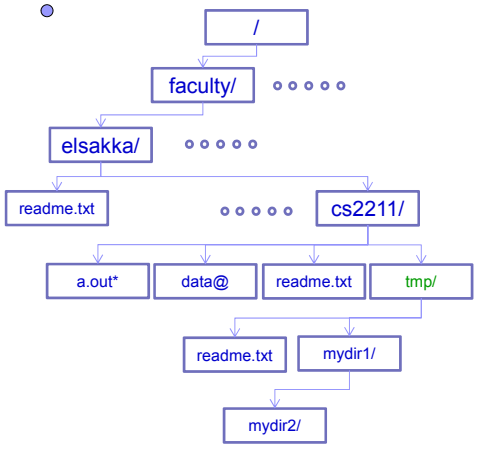
Working with Directories

- Examples:

```
obelix[44] > cd ~/cs2211/tmp;
obelix[45] > ls
mydir1/     readme.txt
obelix[46] > ls mydir1
mydir2/
obelix[47] > rmdir mydir1/mydir2
obelix[48] > ls mydir1
obelix[49] > rmdir mydir1
```

How can I remove mydir1?

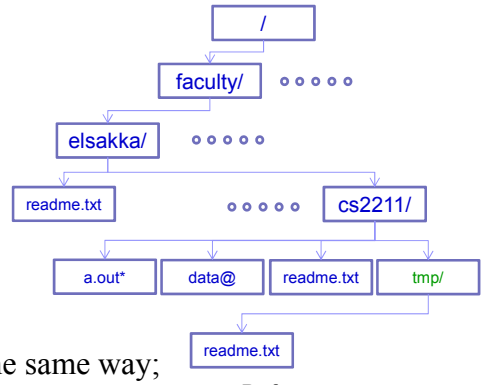
Now, mydir1 becomes empty, so rmdir will work fine



Working with Directories

- Move a file from one directory to another

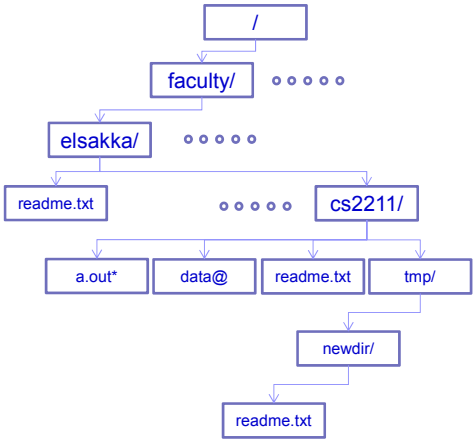
```
obelix[50] > cd ~/cs2211/tmp
obelix[51] > ls
readme.txt
obelix[52] > mkdir newdir
obelix[53] > ls
newdir/     readme.txt
obelix[54] >  mv readme.txt newdir
obelix[55] >  ls
newdir/
obelix[56] >  ls newdir
readme.txt
```



Before

- You can also move a directory the same way; after all, it is just a special file

Working with Directories

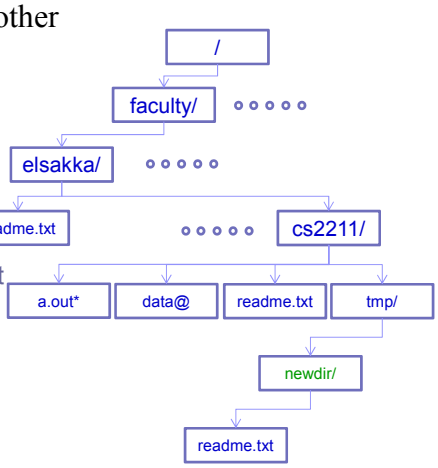


After

Working with Directories

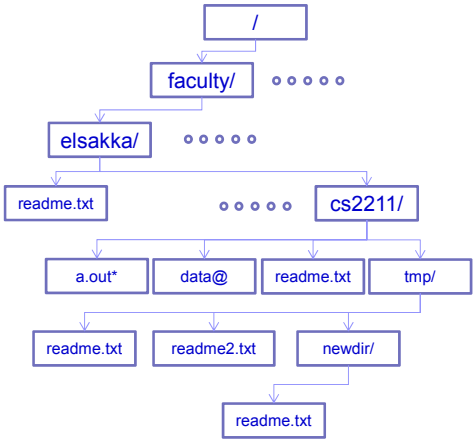
- Copy a file from one directory to another

```
obelix[57] > cd ~/cs2211/tmp/newdir
obelix[58] > ls
readme.txt
obelix[59] > ls ..
newdir/
obelix[60] > cp readme.txt  ..
obelix[61] > cp readme.txt  ../readme2.txt
obelix[62] > ls
readme.txt
obelix[63] > ls ..
newdir/   readme.txt   readme2.txt
```



Before

Working with Directories



After

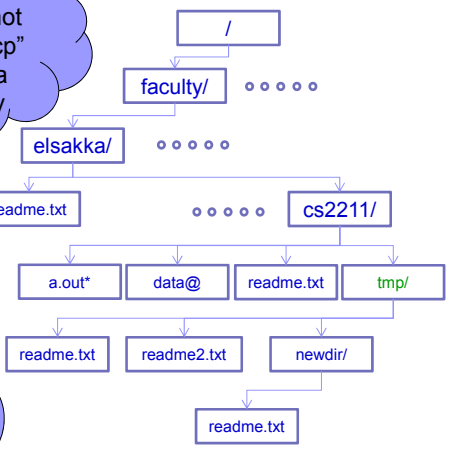
Working with Directories

- Copying a directory

```
obelix[64] > cd ~/cs2211/tmp
obelix[65] > ls
newdir/  readme.txt  readme2.txt
obelix[66] > cp newdir newdir2
cp: newdir: is a directory
obelix[67] > cp -r newdir newdir2
obelix[68] > ls
newdir/  newdir2/ readme.txt  readme2.txt
obelix[69] > ls newdir
readme.txt
obelix[70] > ls newdir2
readme.txt
```

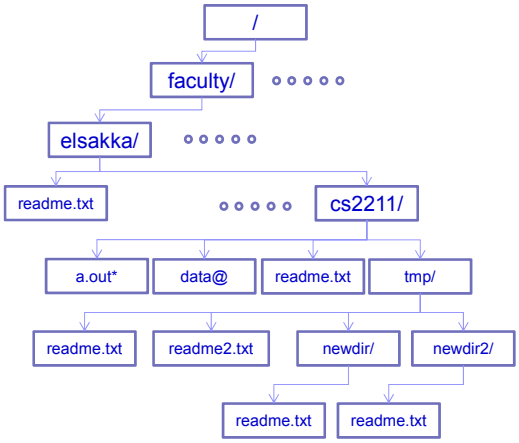
You can not use just "cp" to copy a directory

You must do a recursive copy "cp -r" to copy a directory



Before

Working with Directories



After

Working with Directories

- Some shells (including csh and tcsh, and bash, *but not* sh) provide pushd and popd directory commands
  - pushd changes directories, but remembers the previous one by pushing it on to a *stack*
  - popd changes directories back to the last directory placed on the *stack* by pushd

```
obelix[71] > cd ~/cs2211/tmp
obelix[72] > pushd newdir
~/cs2211/tmp/newdir   ~/cs2211/tmp
obelix[73] > pwd
/faculty/elsakka/cs2211/tmp/newdir
obelix[74] > cd /
obelix[75] > pwd
/
obelix[76] > popd
~/cs2211/tmp
obelix[77] > pwd
/faculty/elsakka/cs2211/tmp
```

Pushed directory

New current directory

Working with Directories

- To keep track of what you have so far in the stack, use dirs command

```
obelix[78] > cd ~/cs2211/tmp
obelix[79] > pushd newdir
~/cs2211/tmp/newdir   ~/cs2211/tmp
obelix[80] > pwd
/faculty/elsakka/cs2211/tmp/newdir
obelix[81] > cd /usr
obelix[82] > pushd ~
~   /usr  ~/cs2211/tmp
obelix[83] > cd /tmp
obelix[84] > dirs
/tmp   /usr  ~/cs2211/tmp
obelix[85] > dirs -v -l
0          /tmp
1          /usr
2          /faculty/elsakka/cs2211/tmp
```

dirs means directory stack most recent to the left first

You got the full pathname due to "-l"

You got 0, 1, and 2, due to "-v"

Working with Directories

- What if you need to locate a file, or a set of files, in a large directory structure?
  - Using cd and ls would be very tedious!
- The command find is used to search through directories to locate files
  - Wildcards can be used
    - when the exact file name is unknown, or
    - to find multiple files at once
  - Files can also be found based on size, owner, creation time, type, permissions, and so on
  - You can also automatically execute commands on each file found
- Extra optional reading:
  If you are interested to know more about find, you may read "man -s 1 find" or read the text book (pp. 750-763)
  - The "-s 1" is specified, as there are more than one find in the man pages (See man -f find)

More Files and Directories

- What files do I already have in my home directory?
  - Startup files (e.g., .login, .cshrc)
    - Contain commands that run after you successfully enter your password, but before you get a prompt
- Assume you have not used your account before

```
obelix[1] > ls
folder/
obelix[2] >
```

  - Where are the rest of the files?
    - In Unix, files begin with a '*dot*' are hidden files
  - Use ls with -a option to see all files

```
obelix[3] > ls -a
./              .alias.sun4@    .emacs*     .profile*     .xinitrc.sun*
../             .alias.sun4m*   .forward    .solregis/    .xsession*
.Xdefaults*     .alias.sun4u@   .login*     .tcshrc*      folder/
.alias.rs6000*  .cshrc*         .mwmrc*     .twmrc*
```


More Files and Directories

- If a dot file copied/moved to a file name that does not start with a dot, the copied/moved file will not be hidden

```
obelix[4] > cd ~
obelix[5] > ls
folder/
obelix[6] > cp .login abc
obelix[7] > ls
abc     folder/
```

More Files and Directories

- cp - copy files

```
obelix[8] > cp .login abc
obelix[9] > cp -i .login abc
cp: overwrite abc (yes/no)? y
```

the copy will be performed, even if the target file exist

The -i option (interactive) prompts for confirmation whenever the copy would overwrite an existing target

- For more details, read "man cp"

More Files and Directories

- rm - remove files

```
obelix[10] > ls
abc
obelix[11] > rm -i abc
rm: remove abc (yes/no)? n
obelix[12] > ls
abc    folder/
```

The -i option for confirmation before removing any files

abc file still exit, since I did not confirm the removal

- For more details, read "man rm"

More Files and Directories

- head command displays the first few lines of a file

```
obelix[13] > head -5 abc
#
#  WGUI is twm or mwm
#

if (!($?HOSTTYPE)) then
obelix[14] > head -5 .login
#
#  WGUI is twm or mwm
#

if (!($?HOSTTYPE)) then
```

- For more details, read "man head"

More Files and Directories

- tail command displays the last few lines of a file

```
obelix[15] > tail -10 abc
#
 case sun3:
   setenv WGUI /usr2/lib/X11R4/bin/twm
 breaksw
#
 default:
   echo "** .login: Unknown Host Type **"
 breaksw

endsw
obelix[16] >
```

- For more details, read "man tail"