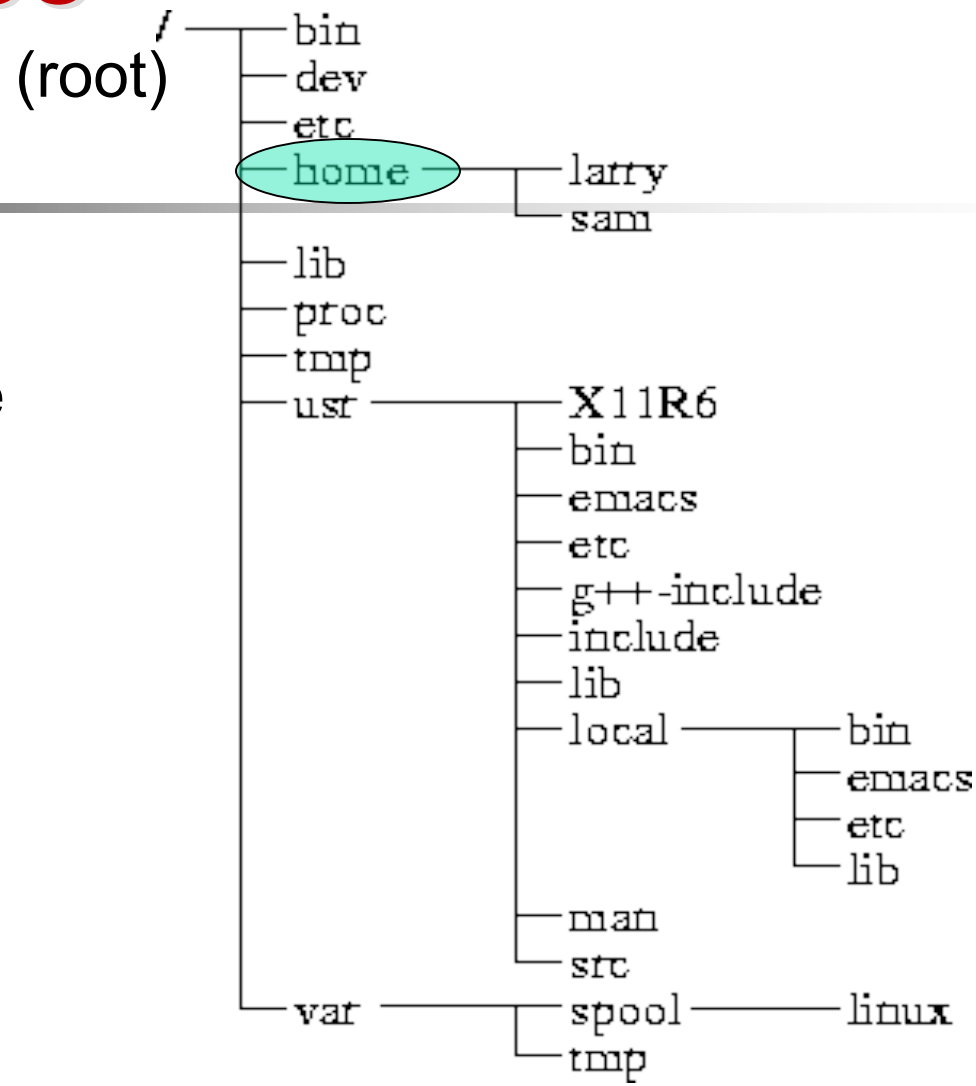
# Linux

The Linux operating system (OS) was first coded by a Finnish computer programmer called **Linus Benedict Torvalds** in 1991, when he was just 21! He had got a new 386, and he found the existing DOS and UNIX too expensive and inadequate.

In those days, a UNIX-like tiny, free OS called **Minix** was extensively used for academic purposes. Since its source code was available, Linus decided to take Minix as a model.

# File management

# Directory Tree



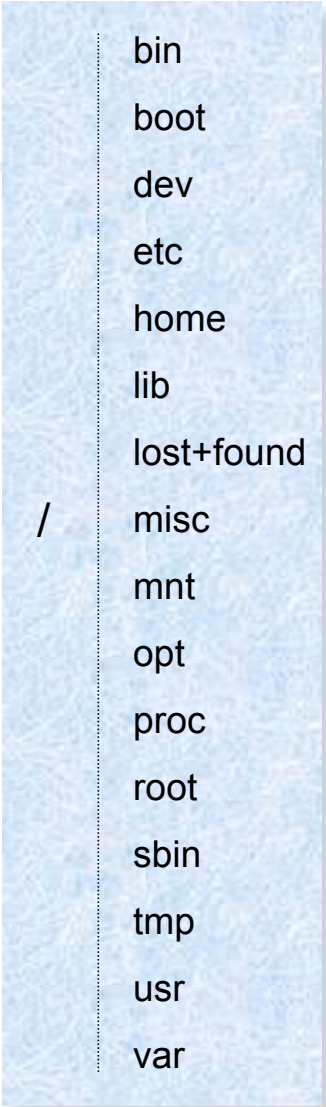

When you log on the the Linux OS using your username you are automatically located in your home directory.

# Home directory

- You can see what your home directory is called by entering
- pwd (print current working directory)

# Linux directories

- **/bin** System binaries, including the command shell
- **/boot** Boot-up routines
- **/dev** Device files for all your peripherals
- **/etc** System configuration files
- **/home** User directories
- **/lib** Shared libraries and modules
- **/lost+found** Lost-cluster files, recovered from a disk-check
- **/mnt** Mounted file-systems
- **/opt** Optional software
- **/proc** Kernel-processes pseudo file-system
- **/root** Administrator's home directory
- **/sbin** System administration binaries
- **/usr** User-oriented software
- **/var** Various other files: mail, spooling and logging



# Subdirectories

| | |
|---|---|
| /root | directory, starting point of the directory tree |
| /home | (private) directories of users |
| /devDevice | files that represent hardware components |
| /etc | Important files for system configuration |
| /etc/init.d | Boot scripts/usr/binGenerally accessible programs |

# File handling commands

chmod

cp

file

mv

rm

head

tail

cat

# Some important commands in Linux

- **`ls`**, Give a listing of the current directory. Try also `ls -l`
- **`cp`**, Copy file from source to destination
- **`mv`**, Move file from source to destination. If both are the same directory, the file is renamed
- **`vi`**, Edit a file. `vi` is one of the most powerful text editors
- **`chmod`**, Change file permissions
- **`mkdir`**, **`rmdir`** Make/Remove a directory
- **`cd`**, Change directory
- **`rm`**, Remove a file. Can also remove directory tree
- **`man ls`**, Get help for `ls`. All commands have help

Ex:

$ ls | more

$ ls > dir_listing.txt

$ cat < file.sh

append:

$ ls >> dir_listing.txt

The following adds the contents of File1 at the end of File2:

$ cat File1 >> File2

# Copy a file/dir to another file/dir

| | |
|---|---|
| `cp file1 file2` | Copy to the same directory and change filename |
| `cp file1 ../dir/file2` | Copy to different directory and change filename |
| `cp file1 ../dir/.` | Keep the same filename |
| | |
| `cp –r dir1 dir2` | Copy directory recursively |
| `cp –r dir1 new_dir/dir2` | Copy directory recursively to another directory |
| | |
| `cp –p file3 file4` | Preserve the modification time and permission modes |

## Directory Related Commands:

cd

mkdir

pwd

ls

## Process Related Commands:

ps -e

top

netstat

pstree

kill

**CTRL ALT Remove (or Delete)**

## Disk related commands:

du - Summarize disk usage of each FILE, recursively for directories.

df - report filesystem disk space usage

```
du -hs /home/username
```

```
df -h
```

“h” for human readable

# Wildcards

- You can substitute the * as a wildcard symbol for any number of characters in any filename.
- If you type just * after a command, it stands for all files in the current directory:

  ***lpr ****  will print all files

- You can mix the * with other characters to form a search pattern:

  ***ls a*.txt***  will list all files that start with “a” and end in “.txt”

- The “**?**” wildcard stands for any single character:

  ***cp draft?.doc***  will copy *draft1.doc*, *draft2.doc*, *draftb.doc*, etc.

# Help on command line

- man : Type ***man*** and the name of a command to read the manual page for that command. e.g. “man ls”
- apropos: gives a list of commands that contain a given keyword in their man page header: e.g. “apropos ls”

# Exercise 1:

- ls
- mkdir phys381
- ls
- cd phys381
- ls
- mkdir Labs
- mkdir Assignments
- mkdir Midterm
- mkdir Final
- ls
- ls -l

?

# Exercise 2:

**Go to your "Labs" Directory and create directories:**
**Lab1, Lab2, Lab3**
**Lab4, Lab5, Lab6**

# chmod

Change the access mode of one or more files

```
chmod u+x file1
chmod go-w file2

chmod u=rwx, g=rx, o=x file3
chmod 751 file3
```

→ Same as above, 7=rwx, 5=rx, 1=x

```
chmod =r file4
chmod 444 file4
```

→ Same as above, 4=r, 2=w, 1=x

# tar

Archive files and directories

Create a single file with extension .tar

```
tar –cvf file123.tar file1 file2 file3
```

Create archive file named file123.tar in verbose mode with contents, file1, file2, and file3

```
tar –xvf file123.tar
```

Expand file123.tar in verbose mode and generate the original files and directories back

# When an application freezes

With the ***ps*** command (*ps* stands for "process status") you find out the identity of the program you want to get rid of.

Then ***kill*** will finish it off.

To display all processes owned by the current user type

*ps ux* and hit return: $ ps ux

This will show a listing of processes similar to:

```
 USER PID %CPU %MEM VSZ RSS TTY STAT START TIME COMMAND
ouyed 3064 0.1 3.6 18324 9088 ? S 17:55 0:00 /usr/bin/gnome-session
ouyed 3130 0.0 0.6 3584 1696 ? S 17:55 0:00 xscreensaver -nosplash
ouyed 3137 0.0 2.4 16296 6172 ? S 17:55 0:00 magicdev --sm-client-id default4
ouyed 3141 0.0 2.8 16676 6936 ? S 17:55 0:00 eggcups --sm-client-id default6
ouyed 3216 0.2 3.1 12788 7844 ? S 17:56 0:00 emacs
```

# The *kill* Command

Now, if you want to terminate for example the emacs process you would look up the process identifier (PID) in the above table (3216), and say:

$ kill -9 3216

The *-9* will ensure "execution".

A convenient short cut is the Alt-Ctrl-Esc key combination, which allows you to simply click on the application you want to kill.

**If you know the "Task-Manager" which pops up under Windows NT/2000/XP when you press CTRL+ALT+DEL then you know what "*ps" command* does on Linux.**

# File Transfer

SCP : Secure Copy

To copy a file from remote machine to your system:

scp student@machine.pjl.ucalgary.ca:/home/student/sudentfile.txt .

To copy file from your machine to remote:

scp myfile.txt student@machine.pjl.ucalgary.ca:/home/student/

VIEWING PDF Files

$ xpdf filename.pdf

VIEWING GIF/PNG/JPEG Files

$ xview filename.gif

PRINTING

$ lpr filename.pdf

CHECKING PRINTER STATUS

$ lpq

DELETING YOUR PRINTER JOB

$ lprm job#

# Emacs

**Phys 381**

R. Ouyed

# Why learn a text editor?

- Text editors are the lifeblood of programming
- You'll need to be able to easily cut, paste, import, modify, automate text
- You need to be able to do this on any environment, even unix/linux

# Why Emacs?

- Not the only good choice
- (VI, VIM are other good options)
- Works on many platforms
- Works with or without a GUI
- Extremely powerful

# Why Emacs (cont…)?

**emacs** is one of the most popular editors available to UNIX programmers.

This tutorial will not teach you every **emacs** command that you will ever need to know. Rather, the goal of this tutorial is to introduce you to some of the key commands that are available.

# What is Emacs exactly?

- **Emacs is a powerful and extensible text editor.**
  - **Emacs originally was an acronym for Editor MACroS.**
  - **The first Emacs was a set of macros written in 1976 at MIT by Richard M Stallman.**
  - **Emacs has become a standard editor used by programmers worldwide.**

# There is more ...

- Content sensitive major modes for a wide variety of file types, from plain text to source code to HTML files, with syntax coloring.
- Complete built-in documentation, including a tutorial for new users.
- Highly extensible through the Emacs Lisp language.
- Support for many languages and their scripts, including all the European ᚷ Latinᛇ scripts, Russian, Greek, Japanese, Chinese, Korean, Thai, Vietnamese, Lao, Ethiopian, and some Indian scripts. (Sorry, Mayan hieroglyphs are not supported.)
- A large number of extensions which add other functionality. The GNU Emacs distribution includes many extensions; many others are available separately, even a web browser

# Creating and Opening Files

- To create a file, type the name of the file that you wish to create after the emacs command. Example:

**emacs phys381.f90 &**

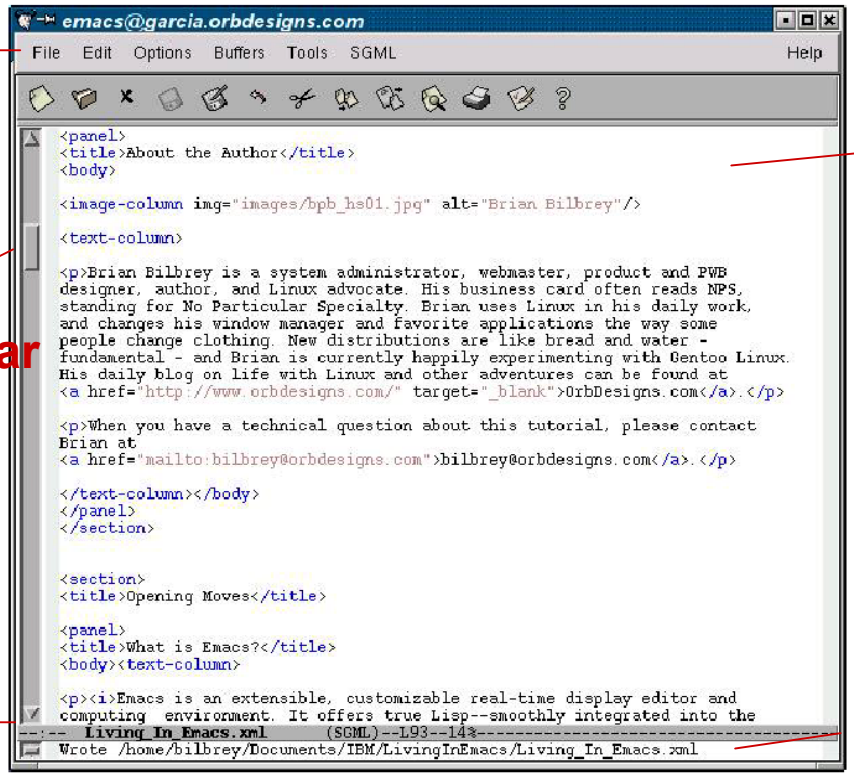
**This is important! Always add it !!!**

- To open a file, type the name after the command, just as above.

# The Emacs Display

- The display in **Emacs** is divided into three basic areas.
  - The top area is called the text window. The text window takes up most of the screen, and is where the document being edited appears.
  - Below the text window, there is a single mode line (in reverse type). The mode line gives information about the document, and about the **Emacs** session.
  - The bottom line of the **Emacs** display is called the minibuffer. The minibuffer holds space for commands that you give to **Emacs**, and displays status information.

# The Emacs view




**Menu**

**Editing windows**

**Mouse-enabled scroll bar**

**Status bar**

**Mini buffer**

# Information Bar

**A look at the Information Bar**

```
----:**-F1  test.cpp       All (2,0)      (C++/l Abbrev)------------------------
```

The `**` means that the file has been edited and is unsaved. When saved, they will change to two hyphens (`--`).

`test.cpp` is the file name.

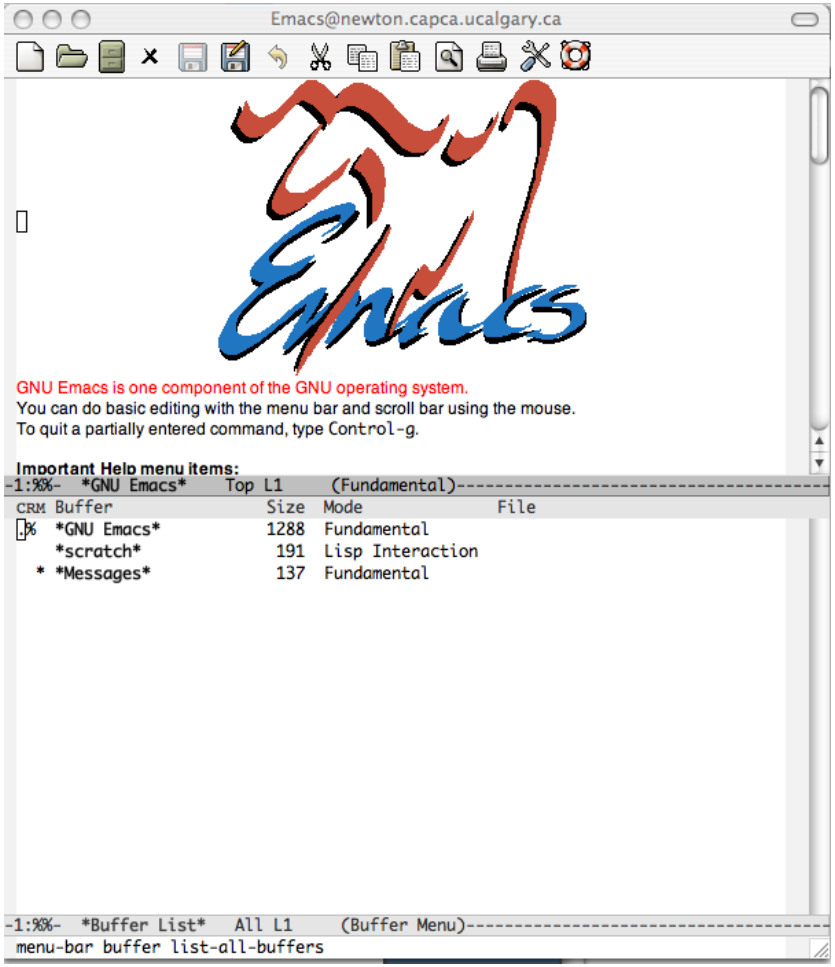
`(2,0)` is the row number and column number of the cursor position.

`(C++/l Abbrev)` is the mode that emacs is in for the currently open file. For this example, it's telling us that emacs is in the C++ mode, therefore it will be applying any known formatting and syntax highlighting rules.

The blank second row of the information bar is the minibuffer. When you perform an action in emacs, information about it or your current focus can appear here. For example, if when you open a file from within emacs (as you will see below), the phrase `Find file: ~/` will appear here and when you start to type a file name, it will appear here instead of in the file buffer.

# The Buffer

# Buffers in action



# Buffer(s)

The temporary space that is used to edit file information before it is written to the file.

# Buffers in action

- Your listing should resemble this:

```
 MR Buffer          Size Mode          File
 -- ------          ---- ----          ----
.%  *GNU Emacs*   1288  Fundamental
*   *scratch*      191  Lisp Interaction
*   *Messages*     137  Fundamental
```

# Modes

Text mode
Fortran mode
Latex mode
Gnuplot mode

# Modes

- Modes are the methods by which Emacs features are expressed in the context of specific types of content.
- That is, indenting behaves differently in a Fortran source code file than in an Tex file or in a letter to your boss.
- There are two different types of modes: major and minor.

# Text Mode

# Practise Exercise

1. Open an xterm window if necessary.
2. At the command-line prompt, type "**emacs phys381-emacs.txt &**" and press Enter.

You are now in edit mode and any keys that you type will be inserted into the current buffer (file). In your phys381-emacs.txt type:

```
 1.0   3.0
 0.4   20.0
 3.2   1.2

-0.1   2.1
 31.0 -23.0
-12.1 3.6
```

Separated by 2 lines!

Then save your file (type "ls -l" in your directory to check the size of your file)

# Gnuplot Mode

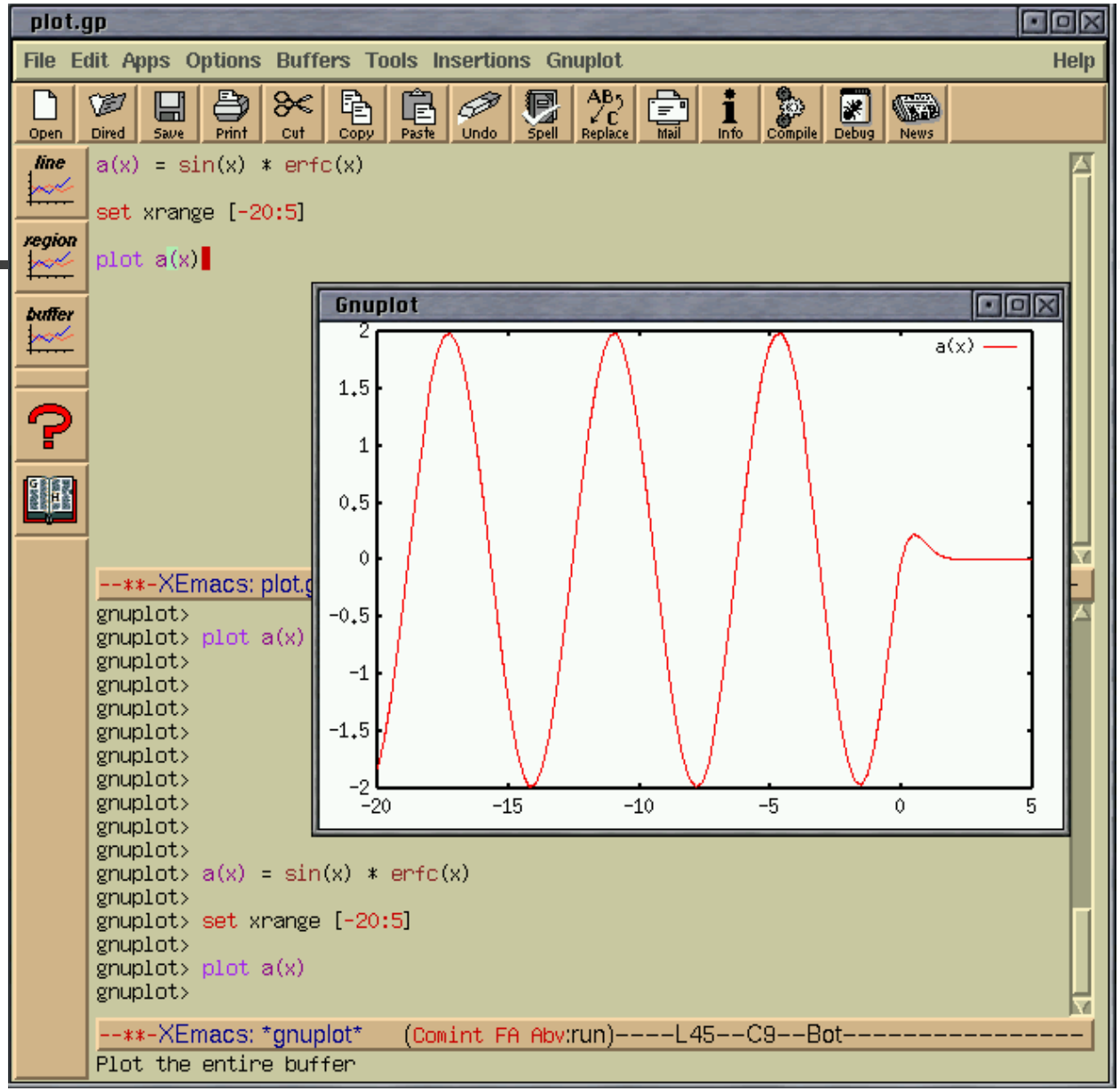
# Practise Exercise

At the command-line prompt, type "**emacs phys381-emacs.gp &**" and press Enter.

You are now in GNUPLOT mode and any keys that you type will be inserted into the current buffer (file). In your phys381-emacs.gp type:

```
set terminal X11
set multiplot
plot "phys391-emacs.txt" index 0 using 1:2 with points, \
     "phys391-emacs.txt" index 1 using 1:2 with lines
unset multiplot
reset
```

Then save your file (type "ls -l" in your directory to check the size of your file)


plot.gp
File Edit Apps Options Buffers Tools Insertions Gnuplot Help
Open Dired Save Print Cut Copy Paste Undo Spell Replace Mail Info Compile Debug News
line
region
buffer
a(x) = sin(x) * erfc(x)
set xrange [-20:5]
plot a(x)
Gnuplot
a(x)
2
1.5
1
0.5
0
-0.5
-1
-1.5
-2
-20
-15
-10
-5
0
5
--**-XEmacs: plot.g
gnuplot>
gnuplot> plot a(x)
gnuplot> a(x) = sin(x) * erfc(x)
gnuplot> set xrange [-20:5]
gnuplot> plot a(x)
--**-XEmacs: *gnuplot* (Comint FA Abv:run)----L45--C9--Bot----------
Plot the entire buffer

# Latex Mode

# Latex in Emacs

# AUCTeX

- Created in 1992 by students at Aalborg University Center (Denmark), hence the name AUCTEX; now mantained by David Kastrup.
- AUCTEX is a "sophisticated TEX environment for GNU Emacs".

# AUCTeX

- (X)Emacs has main mode for LaTeX
  - Usually automatically invoked for `.tex`
- AUCTeX: extends standard LaTeX mode
- Best if invoked via `.emacs`
- Many, many nice features

# Example 1

File Edit Options Buffers Tools Preview LaTeX Command Help

```
\documentclass[10pt]{article}

\usepackage[latin1]{inputenc}
\usepackage[T1]{fontenc}
\usepackage[dvips]{graphicx}

\title{TEST DOCUMENT}
\author{\L{}ukasz Grzegorz Maciak}


\begin{document}

\maketitle

\section{Test Section}
```

**1.1 Test Subsection**

**1.1.1 Test Sub-subsection**

```
This is \textbf{bold}, \textit{italic} and \texttt{mono-space}.
```

$$\sum_{x=0}^{12} 2x + \frac{10}{x}$$

```
\end{document}
```

--:-- test.tex (LaTeX Compiling Fly)--L21--C59--All----------

Beginning of buffer

# Example 2

circ.tex

File Edit Options Buffers Tools Preview LaTeX Command Math Ref Help

```
\section{Die gerade Linie}
Wir betrachten hier zunächst nur die gerade Linie im ersten Oktanten,
die durch den Punkt $0 \choose 0$ geht. Alle anderen Linien lassen
sich durch Vertauschen von $x$ und~$y$ sowie Vorzeichenwechsel
erzeugen.  Im ersten Oktanten gilt $x \geq y \geq 0$. Zum Zeichnen
einer Linie genügt es also, $x$ durchlaufen zu lassen und für $y$ die
dazugehörigen Werte zu berechnen und zu runden.

Die Gleichung einer Geraden durch $\Delta x \choose \Delta y$ lautet:
\begin{equation}
\label{lgi}
y = \frac{\Delta y}{\Delta x}x
\end{equation}
%
Nun stellen wir $y$ als Summe eines ganzzahligen Wertes $e$ und eines
gebrochenen Wertes $f$ dar, für den gilt: $-0.5 \leq f < 0.5$.  Somit
stellt dann $e$ den gewünschten, auf die nächste ganze Zahl gerundeten
$y$-Wert dar. Jetzt formen wir (\ref{lgi}) um:
\begin{eqnarray}
e + f &=& x \frac{\Delta y}{\Delta x}\nonumber\\
e \Delta x + f \Delta x &=& x \Delta y\nonumber\\
f \Delta x - \left\lceil\frac{\Delta x}2\right\rceil &=&
x \Delta y - e \Delta x - \left\lceil\frac{\Delta x}2\right\rceil \label{lgii}
\end{eqnarray}
%
Den linken Ausdruck in (\ref{lgii}) bezeichnen wir jetzt mit $d$.  Für
positive gerade Werte von $\Delta x$ ist offensichtlich $d < 0$ eine
zu~$f < 0.5$ equivalente Bedingung.

Für ungerade Werte von~$\Delta x$ ist $f < 0.5$ equivalent zu
$d + 0.5 < 0$.
Da $d$ stets eine ganze Zahl ist, ist dies wieder zu $d < 0$
equivalent.

% A few intentional errors to check error processing:
%
% The following line should flag a PostScript error when previewing
```

-1:-- circ.tex 15% (67,0) (LaTeX/FM Ref Fill)----23:21----

# Example 3

File Edit Options Buffers Tools LaTeX Command Help

```
Most keyboard shortcuts are in fact commands in Emacs. Any command may
be entered by
\begin{quote}
M-x command-name
\end{quote}
Any command is a sequences of words connected by dashes. Eg. the
command to move forward one space is in fact {\bf forward-char}. When
ever you press the right arrow key it actually is a shortcut to this
command. Many commands have a keyboard shortcut, but many
don't.

You'll find that most Emacs shortcuts starts with C-c or C-x. C-x is
reserved for shortcuts which are effective all the time (like C-x C-f
for {\bf find-file} while C-c is reserved for shortcuts which depends
on a particular packes (eg.\ AUCTeX uses C-c C-c to run LaTeX when you
edit a TeX document). A lot of the other chars are used for basic
editing commands like C-f for {\bf forward-char}. These commands are
-1:--  template.tex      [(LaTeX Fill)]--L115--C21--73%-----------------------------
BCKSPACE - but for a long time ago you couldn't be certain that these
keys actually existed on a keyboard (difficult to imagine, I know).

Some of the most basic keyboard shortcuts are
\begin{tabular}{ll}
C-g & Quit (very important) \\
C-x C-f & Open file \\
C-x C-s & Save file \\
C-x C-w & Save file as
\end{tabular}

\subsection{Windows and buffers}

Let us explain some basic terminology. What is usually called a window is in
Emacs called a {\bf frame}. You may only use frames if you use a
X-Window system. A frame may be divided into several {\bf
  windows}. This \htmladdnormallink{
-1:--  template.tex      [(LaTeX Fill)]--L126--C22--89%-----------------------------
```

# Fortran Mode

# Compiling a code

- Type `M-x compile` and the prompt in the mini-buffer reads, `Compile command:`

  Type in

  ```
  gfortran -o hello hello.f90
  ```

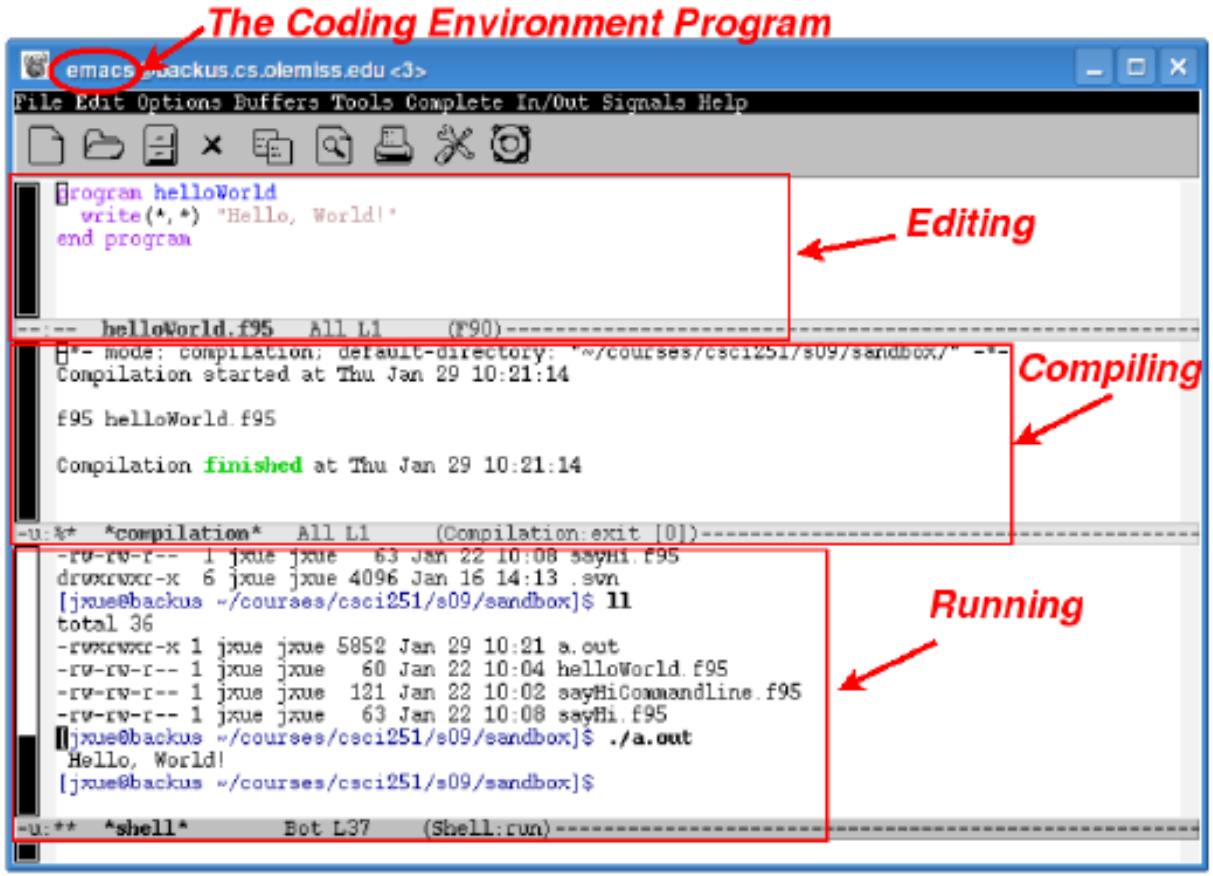
- To see if my program works, I'll run it from inside Emacs:

  ```
  M-! ~/hello.
  ```

  There in the mini-buffer is my output:

  ```
  Hello, World!
  ```

# Compiling a code



# Thank you!