

# CS11002 Programming and Data Structures Spring 2008

## Introduction

Goutam Biswas  
Abhijit Das  
Dipankar Sarkar

Department of Computer Science & Engineering  
Indian Institute of Technology, Kharagpur

Jan 04, 2008

# Syllabus

# Syllabus

- Introduction to digital computers

# Syllabus

- Introduction to digital computers
- Basic programming constructs
  - Variables and simple data types
  - Assignments
  - Input/output
  - Conditions and branching
  - Loops and iteration
  - Iterative searching and sorting algorithms

# Syllabus

- Introduction to digital computers
- Basic programming constructs
  - Variables and simple data types
  - Assignments
  - Input/output
  - Conditions and branching
  - Loops and iteration
  - Iterative searching and sorting algorithms
- Advanced programming constructs
  - Functions and recursion
  - Recursive sorting algorithms
  - Arrays and strings
  - Structures
  - Pointers and dynamic memory allocation

# Syllabus (contd.)

# Syllabus (contd.)

- Performance analysis of programs

# Syllabus (contd.)

- Performance analysis of programs
- Data structures
  - Abstract data types
  - Ordered lists
  - Stacks and queues



# Syllabus (contd.)

- Performance analysis of programs
- Data structures
  - Abstract data types
  - Ordered lists
  - Stacks and queues

**Programming language: C**

# Textbooks and references

# Textbooks and references

Use any standard textbook on **ANSI C**

# Textbooks and references

Use any standard textbook on **ANSI C**

Do **not** use books written on specific C compilers (Turbo C, gcc)

# Textbooks and references

Use any standard textbook on **ANSI C**

Do **not** use books written on specific C compilers (Turbo C, gcc)

- 1 Brian W. Kernighan and Dennis M. Ritchie, *The C Programming Language*, Prentice Hall of India.
- 2 E. Balaguruswamy, *Programming in ANSI C*, Tata McGraw-Hill.
- 3 Byron Gottfried, *Schaum's Outline of Programming with C*, McGraw-Hill.
- 4 P. Dey and M. Ghosh, *Programming in C*, Oxford University Press.

# Textbooks and references

# Textbooks and references

- 5 Seymour Lipschutz, *Data Structures*, Schaum's Outlines Series, Tata McGraw-Hill.
- 6 Ellis Horowitz, Satraj Sahni and Susan Anderson-Freed, *Fundamentals of Data Structures in C*, W. H. Freeman and Company.
- 7 R. G. Dromey, *How to Solve it by Computer*, Prentice-Hall of India.

# Textbooks and references

- 5 Seymour Lipschutz, *Data Structures*, Schaum's Outlines Series, Tata McGraw-Hill.
- 6 Ellis Horowitz, Satraj Sahni and Susan Anderson-Freed, *Fundamentals of Data Structures in C*, W. H. Freeman and Company.
- 7 R. G. Dromey, *How to Solve it by Computer*, Prentice-Hall of India.
- 8 <http://www.facweb.iitkgp.ernet.in/~pds/notes/>



# Marks distribution

# Marks distribution

- Two class tests:  $10 \times 2 = 20$

# Marks distribution

- Two class tests:  $10 \times 2 = 20$
- Mid-semester test: 30

# Marks distribution

- Two class tests:  $10 \times 2 = 20$
- Mid-semester test: 30
- End-semester test: 50

# Marks distribution

- Two class tests:  $10 \times 2 = 20$
- Mid-semester test: 30
- End-semester test: 50
  
- Final marks of a student:  $M = m \times \alpha$ , where

# Marks distribution

- Two class tests:  $10 \times 2 = 20$
- Mid-semester test: 30
- End-semester test: 50
  
- Final marks of a student:  $M = m \times \alpha$ , where
  - $m$  = Total marks obtained in 100, and

# Marks distribution

- Two class tests:  $10 \times 2 = 20$
- Mid-semester test: 30
- End-semester test: 50
  
- Final marks of a student:  $M = m \times \alpha$ , where
  - $m$  = Total marks obtained in 100, and
  - $\alpha$  = Classes attended / Total number of classes.

# Tentative schedule of theory tests



# Tentative schedule of theory tests

- Class Test 1: February 06, 2008 (Wednesday)

# Tentative schedule of theory tests

- Class Test 1: February 06, 2008 (Wednesday)
- Mid-semester Test: February 22–29, 2008 (Friday to Friday)

# Tentative schedule of theory tests

- Class Test 1: February 06, 2008 (Wednesday)
- Mid-semester Test: February 22–29, 2008 (Friday to Friday)
- Class Test 2: April 02, 2008 (Wednesday)

# Tentative schedule of theory tests

- Class Test 1: February 06, 2008 (Wednesday)
- Mid-semester Test: February 22–29, 2008 (Friday to Friday)
- Class Test 2: April 02, 2008 (Wednesday)
- End-Semester Test: April 21–29, 2008 (Monday to Tuesday)

# Tentative schedule for the laboratory

# Tentative schedule for the laboratory

- Lab test 1: February 15–21, 2008 (Friday to Thursday)

# Tentative schedule for the laboratory

- Lab test 1: February 15–21, 2008 (Friday to Thursday)
- Lab Test 2: April 04–10, 2008 (Friday to Thursday)

# Tentative schedule for the laboratory

- Lab test 1: February 15–21, 2008 (Friday to Thursday)
- Lab Test 2: April 04–10, 2008 (Friday to Thursday)
- Marks distribution
  - Lab Test 1: 25
  - Lab Test 2: 35
  - Daily Performance: 40



# Tentative schedule for coverage

# Tentative schedule for coverage

- Before Class Test 1: Until “iterations” (all loop constructs)

# Tentative schedule for coverage

- Before Class Test 1: Until “iterations” (all loop constructs)
- Before MidSem Test: Until “introduction to pointers”

# Tentative schedule for coverage

- Before Class Test 1: Until “iterations” (all loop constructs)
- Before MidSem Test: Until “introduction to pointers”
- Before Class Test 2: Until “linked structures”

## Tentative schedule for coverage

- Before Class Test 1: Until “iterations” (all loop constructs)
- Before MidSem Test: Until “introduction to pointers”
- Before Class Test 2: Until “linked structures”
- Before EndSem Test: Everything

# Contacts

# Contacts

- Instructors

# Contacts

- Instructors

- Sections 1,2: Goutam Biswas, CSE-207, 81437

`goutam@cse.iitkgp.ernet.in`

`http://www.facweb.iitkgp.ernet.in/~goutam/`



# Contacts

- Instructors

- Sections 1,2: Goutam Biswas, CSE-207, 81437

`goutam@cse.iitkgp.ernet.in`

`http://www.facweb.iitkgp.ernet.in/~goutam/`

- Sections 3,4: Abhijit Das, CSE-123, 82350

`abhij@cse.iitkgp.ernet.in`

`http://www.cse-web.iitkgp.ernet.in/~abhij/`

# Contacts

- Instructors

- Sections 1,2: Goutam Biswas, CSE-207, 81437

`goutam@cse.iitkgp.ernet.in`

`http://www.facweb.iitkgp.ernet.in/~goutam/`

- Sections 3,4: Abhijit Das, CSE-123, 82350

`abhij@cse.iitkgp.ernet.in`

`http://www.cse-web.iitkgp.ernet.in/~abhij/`

- Sections 5,6: Dipankar Sarkar, CSE-115, 83492

`ds@cse.iitkgp.ernet.in`

`http://www.facweb.iitkgp.ernet.in/~ds/`

# Contacts

- Instructors

- Sections 1,2: Goutam Biswas, CSE-207, 81437

`goutam@cse.iitkgp.ernet.in`

`http://www.facweb.iitkgp.ernet.in/~goutam/`

- Sections 3,4: Abhijit Das, CSE-123, 82350

`abhij@cse.iitkgp.ernet.in`

`http://www.cse-web.iitkgp.ernet.in/~abhij/`

- Sections 5,6: Dipankar Sarkar, CSE-115, 83492

`ds@cse.iitkgp.ernet.in`

`http://www.facweb.iitkgp.ernet.in/~ds/`

- Course web-page:

`http://www.facweb.iitkgp.ernet.in/~pds/`

`http://www.facweb.iitkgp.ernet.in/~pds/notes/`

# How to write C programs

## Skeleton of a C program

# How to write C programs

## Skeleton of a C program

Include header files

# How to write C programs

## Skeleton of a C program

Include header files

Declare global variables, constants and function prototypes

# How to write C programs

## Skeleton of a C program

Include header files

Declare global variables, constants and function prototypes

Function bodies

# How to write C programs

## Skeleton of a C program

Include header files

Declare global variables, constants and function prototypes

Function bodies

There must be a **main** function in any C program.



# A complete example

```
#include <stdio.h>

#define PI_4_BY_3 4.1887902048

double radius = 10;

double sphereVol ( double r )
{
    return PI_4_BY_3 * r * r * r;
}

main ()
{
    double area;
    area = sphereVol(radius);
    printf("Radius = %lf, volume = %lf.\n", radius, area);
}
```

# The traditional starter

```
#include <stdio.h>

main ()
{
    printf("Hello, world!\n");
}
```

# The traditional starter

```
#include <stdio.h>

main ()
{
    printf("Hello, world!\n");
}
```

This program takes no input, but outputs the string  
“Hello, world!”  
in a line.

# The short-circuit program

```
#include <stdio.h>

main ()
{
    int n;

    scanf ("%d", &n);
    printf ("%d\n", n);
}
```

# The short-circuit program

```
#include <stdio.h>

main ()
{
    int n;

    scanf ("%d", &n);
    printf ("%d\n", n);
}
```

This program accepts an integer as input and outputs the same integer.

# The square finder

```
#include <stdio.h>

main ()
{
    int n;

    scanf ("%d", &n);
    printf ("%d\n", n*n);
}
```

# The square finder

```
#include <stdio.h>

main ()
{
    int n;

    scanf ("%d", &n);
    printf ("%d\n", n*n);
}
```

This program takes an integer  $n$  as input and outputs the square  $n^2$  of  $n$ .

# A faulty reciprocal finder

```
#include <stdio.h>

main ()
{
    int n;

    scanf ("%d",&n);
    printf ("%d\n",1/n);
}
```



# A faulty reciprocal finder

```
#include <stdio.h>

main ()
{
    int n;

    scanf ("%d", &n);
    printf ("%d\n", 1/n);
}
```

The division  $1/n$  is of integers (quotient).

# A faulty reciprocal finder

```
#include <stdio.h>

main ()
{
    int n;

    scanf ("%d", &n);
    printf ("%d\n", 1/n);
}
```

The division  $1/n$  is of integers (quotient).

The format `%d` is for printing integers.

# The correct reciprocal finder

```
#include <stdio.h>

main ()
{
    int n;

    scanf ("%d", &n);
    printf ("%f\n", 1.0/n);
}
```

# Getting started

# Getting started

- Switch on your **monitor**.

# Getting started

- Switch on your **monitor**.
- Switch on your **PC**.

# Getting started

- Switch on your **monitor**.
- Switch on your **PC**.
- Allow the machine to **boot**. Wait until the log in prompt comes.

# Getting started

- Switch on your **monitor**.
- Switch on your **PC**.
- Allow the machine to **boot**. Wait until the log in prompt comes.

- Supply your **log-in** and **password**:

Login: s<nn>

Password: s<nn>

- Here *s* is your section (a for 1, b for 2, and so on)
- <nn> is the number of your PC.

This opens your **window manager** (usually KDE) with **icons**, the **bottom panel**, and so on. You are now ready to start your work.



# Getting started

# Getting started

- Click on the **terminal** icon to open a **shell** (command prompt).

# Getting started

- Click on the **terminal** icon to open a **shell** (command prompt).
- Edit your program (new or already existing) by an editor.

```
emacs myprog.c &
```

# Getting started

- Click on the **terminal** icon to open a **shell** (command prompt).
- Edit your program (new or already existing) by an editor.  
`emacs myprog.c &`
- Write your program in the editor and **save** it.

# Getting started

- Click on the **terminal** icon to open a **shell** (command prompt).
- Edit your program (new or already existing) by an editor.

```
emacs myprog.c &
```

- Write your program in the editor and **save** it.
- Go to the shell and **compile** your program:

```
cc myprog.c
```

If compilation is successful, an **executable** called `a.out` will be created.

# Getting started

- Click on the **terminal** icon to open a **shell** (command prompt).
- Edit your program (new or already existing) by an editor.

```
emacs myprog.c &
```

- Write your program in the editor and **save** it.
- Go to the shell and **compile** your program:

```
cc myprog.c
```

If compilation is successful, an **executable** called `a.out` will be created.

- **Run** your program:

```
./a.out
```

# Getting started

- Click on the **terminal** icon to open a **shell** (command prompt).
- Edit your program (new or already existing) by an editor.

```
emacs myprog.c &
```

- Write your program in the editor and **save** it.
- Go to the shell and **compile** your program:

```
cc myprog.c
```

If compilation is successful, an **executable** called `a.out` will be created.

- **Run** your program:

```
./a.out
```
- Continue your edit-compile-debug-run-debug cycle.

# Getting started



# Getting started

- **Close** all the windows you opened.

# Getting started

- **Close** all the windows you opened.
- **Log out** from your window manager. This leaves you again in the log-in console.

# Getting started

- **Close** all the windows you opened.
- **Log out** from your window manager. This leaves you again in the log-in console.
- Select the item to **shut down** the machine. Wait until the machine completely shuts down.

# Getting started

- **Close** all the windows you opened.
- **Log out** from your window manager. This leaves you again in the log-in console.
- Select the item to **shut down** the machine. Wait until the machine completely shuts down.
- Switch off your monitor.

# Using emacs

# Using emacs

- **emacs** is a powerful text editor.

# Using emacs

- **emacs** is a powerful text editor.
- Run emacs as: `emacs myprog.c &`

# Using emacs

- **emacs** is a powerful text editor.
- Run emacs as: `emacs myprog.c &`
- Type in your program in the text area



# Using emacs

- **emacs** is a powerful text editor.
- Run emacs as: `emacs myprog.c &`
- Type in your program in the text area
- Navigate with mouse and cursor keys

# Using emacs

- **emacs** is a powerful text editor.
- Run emacs as: `emacs myprog.c &`
- Type in your program in the text area
- Navigate with mouse and cursor keys
- Save your file before closing emacs.

# Using emacs

- **emacs** is a powerful text editor.
- Run emacs as: `emacs myprog.c &`
- Type in your program in the text area
- Navigate with mouse and cursor keys
- Save your file before closing emacs.
  - “File -> Save (Current buffer)”

# Using emacs

- **emacs** is a powerful text editor.
- Run emacs as: `emacs myprog.c &`
- Type in your program in the text area
- Navigate with mouse and cursor keys
- Save your file before closing emacs.
  - “File -> Save (Current buffer)”
  - Click the save button (disk)

# Using emacs

- **emacs** is a powerful text editor.
- Run emacs as: `emacs myprog.c &`
- Type in your program in the text area
- Navigate with mouse and cursor keys
- Save your file before closing emacs.
  - “File -> Save (Current buffer)”
  - Click the save button (disk)
  - “File -> Save buffer as” (to another file)

# Using emacs

- **emacs** is a powerful text editor.
- Run emacs as: `emacs myprog.c &`
- Type in your program in the text area
- Navigate with mouse and cursor keys
- Save your file before closing emacs.
  - “File -> Save (Current buffer)”
  - Click the save button (disk)
  - “File -> Save buffer as” (to another file)
- **Save your file once in every 15 minutes.**

# Using gvim

# Using gvim

- **gvim** is another powerful text editor.



# Using gvim

- **gvim** is another powerful text editor.
- Run gvim as: `gvim myprog.c`

# Using gvim

- **gvim** is another powerful text editor.
- Run gvim as: `gvim myprog.c`
- Hit Insert before you start typing matter

# Using gvim

- **gvim** is another powerful text editor.
- Run gvim as: `gvim myprog.c`
- Hit Insert before you start typing matter
- You will exit the insert mode if you hit Insert when you are already in the insert mode

# Using gvim

- **gvim** is another powerful text editor.
- Run gvim as: `gvim myprog.c`
- Hit Insert before you start typing matter
- You will exit the insert mode if you hit Insert when you are already in the insert mode
- Hit Esc to exit insert mode

# Using gvim

- **gvim** is another powerful text editor.
- Run gvim as: `gvim myprog.c`
- Hit Insert before you start typing matter
- You will exit the insert mode if you hit Insert when you are already in the insert mode
- Hit Esc to exit insert mode
- When in doubt, it is safe to hit Esc several times to come back to view mode

# Using gvim

- **gvim** is another powerful text editor.
- Run gvim as: `gvim myprog.c`
- Hit Insert before you start typing matter
- You will exit the insert mode if you hit Insert when you are already in the insert mode
- Hit Esc to exit insert mode
- When in doubt, it is safe to hit Esc several times to come back to view mode
- Navigate with mouse and cursor keys

# Using gvim

- **gvim** is another powerful text editor.
- Run gvim as: `gvim myprog.c`
- Hit Insert before you start typing matter
- You will exit the insert mode if you hit Insert when you are already in the insert mode
- Hit Esc to exit insert mode
- When in doubt, it is safe to hit Esc several times to come back to view mode
- Navigate with mouse and cursor keys
- You need to save the file by clicking on the appropriate icon (disk).

# Using gvim

- **gvim** is another powerful text editor.
- Run gvim as: `gvim myprog.c`
- Hit Insert before you start typing matter
- You will exit the insert mode if you hit Insert when you are already in the insert mode
- Hit Esc to exit insert mode
- When in doubt, it is safe to hit Esc several times to come back to view mode
- Navigate with mouse and cursor keys
- You need to save the file by clicking on the appropriate icon (disk).
- **Save your file once in every 15 minutes.**



# A practice program

```
#include <stdio.h>

char name[100];
int i;

main ()
{
    printf("Hello, may I know your full name? ");
    scanf("%s",name);
    printf("Welcome %s.\n",name);
    printf("Your name printed backward is : ");
    for (i=strlen(name)-1; i>=0; --i)
        printf("%c",name[i]);
    printf("\n");
}
```

# A practice program (corrected)

```
#include <stdio.h>

char name[100];
int i;

main ()
{
    printf("Hello, may I know your full name? ");
    fgets(name,100,stdin);
    name[strlen(name)-1] = '\0';
    printf("Welcome %s.\n",name);
    printf("Your name printed backward is : ");
    for (i=strlen(name)-1; i>=0; --i)
        printf("%c",name[i]);
    printf("\n");
}
```

# Using a web browser

# Using a web browser

- Open a web browser: **mozilla** or **konqueror**.

# Using a web browser

- Open a web browser: **mozilla** or **konqueror**.
- Set a **proxy**:

# Using a web browser

- Open a web browser: **mozilla** or **konqueror**.
- Set a **proxy**:
  - 144.16.192.213:8080
  - 144.16.192.216:7777
  - 144.16.192.245:8080
  - 144.16.192.247:8080

# Using a web browser

- Open a web browser: **mozilla** or **konqueror**.
- Set a **proxy**:
  - 144.16.192.213:8080
  - 144.16.192.216:7777
  - 144.16.192.245:8080
  - 144.16.192.247:8080
- Bypass proxy for local machines.

# Using a web browser

- Open a web browser: **mozilla** or **konqueror**.
- Set a **proxy**:
  - 144.16.192.213:8080
  - 144.16.192.216:7777
  - 144.16.192.245:8080
  - 144.16.192.247:8080
- Bypass proxy for local machines.
- Type in a URL (web address) in the location field



# Using a web browser

- Open a web browser: **mozilla** or **konqueror**.
- Set a **proxy**:
  - 144.16.192.213:8080
  - 144.16.192.216:7777
  - 144.16.192.245:8080
  - 144.16.192.247:8080
- Bypass proxy for local machines.
- Type in a URL (web address) in the location field
  - <http://www.facweb.iitkgp.ernet.in/~pds/>

# Using a web browser

- Open a web browser: **mozilla** or **konqueror**.
- Set a **proxy**:
  - 144.16.192.213:8080
  - 144.16.192.216:7777
  - 144.16.192.245:8080
  - 144.16.192.247:8080
- Bypass proxy for local machines.
- Type in a URL (web address) in the location field
  - <http://www.facweb.iitkgp.ernet.in/~pds/>
  - <http://www.facweb.iitkgp.ernet.in/~pds/notes/>

# Using a web browser

- Open a web browser: **mozilla** or **konqueror**.
- Set a **proxy**:
  - 144.16.192.213:8080
  - 144.16.192.216:7777
  - 144.16.192.245:8080
  - 144.16.192.247:8080
- Bypass proxy for local machines.
- Type in a URL (web address) in the location field
  - <http://www.facweb.iitkgp.ernet.in/~pds/>
  - <http://www.facweb.iitkgp.ernet.in/~pds/notes/>
  - <http://www.facweb.iitkgp.ernet.in/~adas/>

# Using a web browser

- Open a web browser: **mozilla** or **konqueror**.
- Set a **proxy**:
  - 144.16.192.213:8080
  - 144.16.192.216:7777
  - 144.16.192.245:8080
  - 144.16.192.247:8080
- Bypass proxy for local machines.
- Type in a URL (web address) in the location field
  - <http://www.facweb.iitkgp.ernet.in/~pds/>
  - <http://www.facweb.iitkgp.ernet.in/~pds/notes/>
  - <http://www.facweb.iitkgp.ernet.in/~adas/>
  - <http://www.facweb.iitkgp.ernet.in/~adas/course/lab/PDS/Spring06/>

# Using a web browser

- Open a web browser: **mozilla** or **konqueror**.
- Set a **proxy**:
  - 144.16.192.213:8080
  - 144.16.192.216:7777
  - 144.16.192.245:8080
  - 144.16.192.247:8080
- Bypass proxy for local machines.
- Type in a URL (web address) in the location field
  - <http://www.facweb.iitkgp.ernet.in/~pds/>
  - <http://www.facweb.iitkgp.ernet.in/~pds/notes/>
  - <http://www.facweb.iitkgp.ernet.in/~adas/>
  - <http://www.facweb.iitkgp.ernet.in/~adas/course/lab/PDS/Spring06/>
  - <http://sit.iitkgp.ernet.in/~chitta/courses/pds/slides/starter.html>

# Using a web browser

- Open a web browser: **mozilla** or **konqueror**.
- Set a **proxy**:
  - 144.16.192.213:8080
  - 144.16.192.216:7777
  - 144.16.192.245:8080
  - 144.16.192.247:8080
- Bypass proxy for local machines.
- Type in a URL (web address) in the location field
  - <http://www.facweb.iitkgp.ernet.in/~pds/>
  - <http://www.facweb.iitkgp.ernet.in/~pds/notes/>
  - <http://www.facweb.iitkgp.ernet.in/~adas/>
  - <http://www.facweb.iitkgp.ernet.in/~adas/course/lab/PDS/Spring06/>
  - <http://sit.iitkgp.ernet.in/~chitta/courses/pds/slides/starter.html>
  - <http://10.14.0.4/wbcm/>

# Assignments and submissions

# Assignments and submissions

- Click the link on the day's assignment.



# Assignments and submissions

- Click the link on the day's assignment.
- If your assignment is a **PDF** file, save it to your machine.

# Assignments and submissions

- Click the link on the day's assignment.
- If your assignment is a **PDF** file, save it to your machine.
- Use **xpdf** in order to view PDF files.

```
xpdf newassgn.pdf
```

# Assignments and submissions

- Click the link on the day's assignment.
- If your assignment is a **PDF** file, save it to your machine.
- Use **xpdf** in order to view PDF files.

```
xpdf newassgn.pdf
```

- <http://www.facweb.iitkgp.ernet.in/~pds/2007s/notes.html>
- <http://sit.iitkgp.ernet.in/~chitta/courses/pds/slides/>

# Assignments and submissions

- Click the link on the day's assignment.
- If your assignment is a **PDF** file, save it to your machine.
- Use **xpdf** in order to view PDF files.

```
xpdf newassgn.pdf
```

- <http://www.facweb.iitkgp.ernet.in/~pds/2007s/notes.html>
- <http://sit.iitkgp.ernet.in/~chitta/courses/pds/slides/>
- Consult your lab instructor to know how to submit your programs.

# Some useful Unix commands

# Some useful Unix commands

- Create a directory: `mkdir progs`

# Some useful Unix commands

- Create a directory: `mkdir progs`
- Go to a new directory: `cd progs/`

# Some useful Unix commands

- Create a directory: `mkdir progs`
- Go to a new directory: `cd progs/`
- Go to the parent directory: `cd ../`



# Some useful Unix commands

- Create a directory: `mkdir progs`
- Go to a new directory: `cd progs/`
- Go to the parent directory: `cd ../`
- List all files in a directory: `ls -lF`

# Some useful Unix commands

- Create a directory: `mkdir progs`
- Go to a new directory: `cd progs/`
- Go to the parent directory: `cd ../`
- List all files in a directory: `ls -lF`
- View a file: `cat filename`

# Some useful Unix commands

- Create a directory: `mkdir progs`
- Go to a new directory: `cd progs/`
- Go to the parent directory: `cd ../`
- List all files in a directory: `ls -lF`
- View a file: `cat filename`
- Copy a file to another: `cp file1.c file2.c`

# Some useful Unix commands

- Create a directory: `mkdir progs`
- Go to a new directory: `cd progs/`
- Go to the parent directory: `cd ../`
- List all files in a directory: `ls -lF`
- View a file: `cat filename`
- Copy a file to another: `cp file1.c file2.c`
- Copy a file to a directory: `cp file1.c progs/file3.c`

# Some useful Unix commands

- Create a directory: `mkdir progs`
- Go to a new directory: `cd progs/`
- Go to the parent directory: `cd ../`
- List all files in a directory: `ls -lF`
- View a file: `cat filename`
- Copy a file to another: `cp file1.c file2.c`
- Copy a file to a directory: `cp file1.c progs/file3.c`
- Move a file to another: `mv file1.c file2.c`

# Some useful Unix commands

- Create a directory: `mkdir progs`
- Go to a new directory: `cd progs/`
- Go to the parent directory: `cd ../`
- List all files in a directory: `ls -lF`
- View a file: `cat filename`
- Copy a file to another: `cp file1.c file2.c`
- Copy a file to a directory: `cp file1.c progs/file3.c`
- Move a file to another: `mv file1.c file2.c`
- Move a file to a directory: `mv file1.c progs/file3.c`

# Some useful Unix commands

- Create a directory: `mkdir progs`
- Go to a new directory: `cd progs/`
- Go to the parent directory: `cd ../`
- List all files in a directory: `ls -lF`
- View a file: `cat filename`
- Copy a file to another: `cp file1.c file2.c`
- Copy a file to a directory: `cp file1.c progs/file3.c`
- Move a file to another: `mv file1.c file2.c`
- Move a file to a directory: `mv file1.c progs/file3.c`
- Delete a file: `rm filename`