Fundamentals of Computing: Lecture 30

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

October 19, 2009

Summary of last class

Summary of last class

- C programms can take command lines eg

```
$ progname a1 a2 .. an
```

Summary of last class

- C programms can take command lines eg

```
$ progname a1 a2 .. an
```

```
int main(int argc, char **argv)
{
  /* do something */
}
```

- The variable `argc` is one more than total arguments. in the above example $n + 1$ and,

Summary of last class

- C programms can take command lines eg

```
$ progname a1 a2 .. an
```

```
int main(int argc, char **argv)
{
  /* do something */
}
```

- The variable `argc` is one more than total arguments. in the above example $n + 1$ and,
- `argv[0]` is `progname`, `argv[1]` is `a1` etc `argv[n]` is `an` and `argv[n+1]` is `NULL`.

Summary continued `sprintf` and `sscanf`

Summary continued sprintf and sscanf

```
int answer = 42;
char buf[100];
sprintf(buf, "The answer is %d", answer);
```

Summary continued sprintf and sscanf

```
int answer = 42;
char buf[100];
sprintf(buf, "The answer is %d", answer);
```

- `sprintf` takes a character array `buf`, a format string `fmt` and other arguments and behaves like `printf` but prints stuff to the string `buf`.

Summary continued sprintf and sscanf

```
int answer = 42;
char buf[100];
sprintf(buf, "The answer is %d", answer);
```

- `sprintf` takes a character array `buf`, a format string `fmt` and other arguments and behaves like `printf` but prints stuff to the string `buf`.
- `sscanf` takes a character array `buf`, a formats string `fmt` and other arguments and behaves like `scanf` but scans stuff from the string `buf`

Summary continued sprintf and sscanf

```
int answer = 42;
char buf[100];
sprintf(buf, "The answer is %d", answer);
```

- `sprintf` takes a character array `buf`, a format string `fmt` and other arguments and behaves like `printf` but prints stuff to the string `buf`.
- `sscanf` takes a character array `buf`, a formats string `fmt` and other arguments and behaves like `scanf` but scans stuff from the string `buf`

Ex Write a command like program that prints the sum of its command lines.

File input output

File input output

- How does one write into a file ?

File input output

- How does one write into a file ?
- How does one read from a file.

File input output

- How does one write into a file ?
- How does one read from a file.
- What is a file ?

What is a file ?

What is a file ?

- Files provide way to organise data on the hard disk.

What is a file ?

- Files provide way to organise data on the hard disk.
- It can be hard disk of usb stick or cdrom.

What is a file ?

- Files provide way to organise data on the hard disk.
- It can be hard disk of usb stick or cdrom.
- In Unix everything (almost) is a file.

What is a file ?

- Files provide way to organise data on the hard disk.
- It can be hard disk of usb stick or cdrom.
- In Unix everything (almost) is a file.

File system and directories

File system and directories

- Files are organised into directores

File system and directories

- Files are organised into directores
- Directories are trees (rose trees) and Files are the leaves.

File system and directories

- Files are organised into directores
- Directories are trees (rose trees) and Files are the leaves.
- The root directory is denoted by /.

File system and directories

- Files are organised into directores
- Directories are trees (rose trees) and Files are the leaves.
- The root directory is denoted by `/`.
- A file or directory is completely specified by its path from the root eg `/foo/bar/biz` means the `biz` file under the `bar` directory under the `foo` directory under the root.

File system and directories

- Files are organised into directores
- Directories are trees (rose trees) and Files are the leaves.
- The root directory is denoted by `/`.
- A file or directory is completely specified by its path from the root eg `/foo/bar/biz` means the `biz` file under the `bar` directory under the `foo` directory under the root.
- You can also give relative path eg `foo/bar` means the `bar` directory (or file) under the foo directory under the *current* directory.

File system and directories

- Files are organised into directores
- Directories are trees (rose trees) and Files are the leaves.
- The root directory is denoted by `/`.
- A file or directory is completely specified by its path from the root eg `/foo/bar/biz` means the `biz` file under the `bar` directory under the `foo` directory under the root.
- You can also give relative path eg `foo/bar` means the `bar` directory (or file) under the foo directory under the *current* directory.
- The special names `.` (dot) and `..` means this director and previous (or parent) directory.

Using a file

Using a file

- Open a file

```
FILE *fopen(char *filename, char *mod);
```

Using a file

- Open a file

```
FILE *fopen(char *filename, char *mod);
```

- Use it

```
int getc(FILE*);
int fputc(int c,FILE *);
int fprintf(FILE *fp, char *fmt, ...);
int fscanf(FILE *fp, char *fmt, ...);
```

Using a file

- Open a file

```
FILE *fopen(char *filename, char *mod);
```

- Use it

```
int getc(FILE*);
int fputc(int c,FILE *);
int fprintf(FILE *fp, char *fmt, ...);
int fscanf(FILE *fp, char *fmt, ...);
```

- Close it. int fclose(FILE *);

```
#include <stdio.h>
int main(int argc, char **argv)
{
  int c;
  FILE *fp;
  for(int i = 1; i < argc; i++){
    fp = fopen(argv[i], "r");
    if( fp == NULL){
      fprintf(stderr, "%s: cannot open %s\n", argv[0],argv
      continue;
    }
    while( (c = getc(fp)) != EOF ){
      putchar(c);
    }
    fclose(fp);
  }
  return 0;
}
```