Fundamentals of Computing: Lecture 31

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

October 21, 2009

Summary of last class

Summary of last class

- Files and directories

Summary of last class

- Files and directories
  - Files are collection of data
  - In Unix almost every thing is a filed
  - Files are organised into directories.

Summary of last class

- Files and directories
  - Files are collection of data
  - In Unix almost every thing is a filed
  - Files are organised into directories.
- Operations on a file (open, read/write, close)

```
#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;
}
```

The `FILE` type

The `FILE` type

- A file is represented using a pointer to a `FILE` structure.

The `FILE` type

- A file is represented using a pointer to a `FILE` structure.
- All operations on a file take this pointer as argument.

The `FILE` type

- A file is represented using a pointer to a `FILE` structure.
- All operations on a file take this pointer as argument.
- The exact fields of the `FILE` structure are not relevant to us. It is `FILE *` that is interesting form the C programmers perspective.

Opening a file

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

- The function returns a `FILE *`.
- On error returns a null pointer.
- The mode parameter has the following interpretation
  - "r" means read. If the file does not exists `fopen` returns NULL.
  - "w" means write. If the file exists then truncates it.
  - "a" write at the end of the file. The contents are kept, and file created if it does not exist.
- You can also give "rw" for read and write.
- For more details type `man fopen`

Standard idiom of opening files

```
FILE *fp;
if( (fp = fopen("foo/bar/biz", "r") == NULL )
{
  /* File does not exists or some error has occured.
  Handle it*/
}else {
  /* do some some thing useful with the file */
}
```

As far as C is concerned, a file is a sequence of characters.

As far as C is concerned, a file is a sequence of characters.

The functions `fgetc` and `fputc`

```
int fgetc(FILE *infp);
```

- `fgetc` reads a character for the the file `infp`.

As far as C is concerned, a file is a sequence of characters.

The functions `fgetc` and `fputc`

```
int fgetc(FILE *infp);
```

- `fgetc` reads a character for the the file `infp`.
- `infp` should have been opened in read mode.

As far as C is concerned, a file is a sequence of characters.

The functions fgetc and fputc

```
int fgetc(FILE *infp);
```

- fgetc reads a character for the the file infp.
- infp should have been opened in read mode.
- The value returned by fgetc is EOF if end of file is reached.

As far as C is concerned, a file is a sequence of characters.

The functions fgetc and fputc

```
int fgetc(FILE *infp);
```

- fgetc reads a character for the the file infp.
- infp should have been opened in read mode.
- The value returned by fgetc is EOF if end of file is reached.

```
int fputc(int c, file *outfp);
```

As far as C is concerned, a file is a sequence of characters.

The functions fgetc and fputc

```
int fgetc(FILE *infp);
```

- fgetc reads a character for the the file infp.
- infp should have been opened in read mode.
- The value returned by fgetc is EOF if end of file is reached.

```
int fputc(int c, file *outfp);
```

- fputc writes the character corresponding to c in outfp.

As far as C is concerned, a file is a sequence of characters.

The functions fgetc and fputc

```
int fgetc(FILE *infp);
```

- fgetc reads a character for the the file infp.
- infp should have been opened in read mode.
- The value returned by fgetc is EOF if end of file is reached.

```
int fputc(int c, file *outfp);
```

- fputc writes the character corresponding to c in outfp.
- outfp should have been opened in write or append mode.

Standard idiom to use `fgetc` and `fputc`

```
void copy(FILE *infp, FILE *outfp)
{
  int c;
  while( (c = fgetc(infp)) != EOF) fputc(c, outfp);
}
```

The function `fscanf` and `fprintf`

```
int fscanf(FILE *infp, char *fmt,...);
int fprintf(FILE *outfp, char *fmt, ...);
```

Same as `scanf` and `printf` but uses files

Files opened at the start of the program

- The files `FILE *stdin, stdout, stderr` are open when the program starts.
- As the name suggests `stdin` is the input, `stdout` is the output and `stderr` is for sending error messages.

eg. `printf("%d %c",x,y)` is equivalent to
`fprintf(stdout,"%d %c",x,y)`

Why do we need `stderr`

cat revisited

```
#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;
}
```