# File Management in C

## Console oriented Input/Output

- Console oriented – use terminal (keyboard/screen)
- scanf(“%d”,&i) – read data from keyboard
- printf(“%d”,i) – print data to monitor
- Suitable for small volumes of data

# Real-life applications

- Large data volumes
- E.g. physical experiments (CERN collider), human genome, population records etc.
- Need for flexible approach to store/retrieve data
- Concept of *files*

# Files

- File – place on disc where group of related data is stored
  - E.g. your C programs, executables

- High-level programming languages support file operations
  - Naming
  - Opening
  - Reading

## Defining and opening file

- To store data file in secondary memory (disc) must specify to OS
  - Filename (e.g. sort.c, input.data)
  - Data structure (e.g. FILE)
  - Purpose (e.g. reading, writing, appending)

# Filename

- String of characters that make up a valid filename for OS

- May contain two parts
  - Primary
  - Optional period with extension

# General format for opening file

```
FILE *fp;  /*variable fp is pointer to type FILE*/

fp = fopen("filename", "mode");
/*opens file with name filename , assigns identifier to fp */
```

- fp
  - contains all information about file
  - Communication link between system and program
- Mode can be
  - **r** open file for reading only
  - **w** open file for writing only
  - **a** open file for appending (adding) data

## Different modes

- Writing mode
  - if file already exists then *contents are deleted,*
  - else new file with specified name created
- Appending mode
  - if file already exists then file opened with contents safe
  - else new file created
- Reading mode

```
FILE *p1, *p2;
p1 = fopen(“data”,”r”);
p2= fopen(“results”, “w”);
```

  - if file already exists then opened with contents safe
  - else error occurs.

## Additional modes

- r+  open to beginning for both reading/writing

- w+  same as w except both for reading and writing

- a+   same as ‘a’ except both for reading and writing

# Closing a file

- File must be closed as soon as all operations on it completed

- Ensures
  - All outstanding information associated with file flushed out from buffers
  - All links to file broken
  - Accidental misuse of file prevented

- If want to change mode of file, then first close

# Closing a file

Syntax: **fclose**(file_pointer);

Example:

```
FILE *p1, *p2;
p1 = fopen(“INPUT.txt”, “r”);
p2 =fopen(“OUTPUT.txt”, “w”);
……..
……..
fclose(p1);
fclose(p2);
```

- pointer can be reused after closing

## Input/Output operations on files

- C provides several different functions for reading/writing

- getc() – read a character
- putc() – write a character
- fprintf() – write set of data values
- fscanf() – read set of data values
- getw() – read integer
- putw() – write integer

# Program to read/write using getc/putc

```
#include <stdio.h>
main()
{           FILE *fp1;
            char c;
            f1= fopen(“INPUT”, “w”);   /* open file for writing */

            while((c=getchar()) != EOF)           /*get char from keyboard until
CTL-Z*/
                        putc(c,f1);                               /*write a
character to INPUT */

            fclose(f1);                                           /* close INPUT
*/
            f1=fopen(“INPUT”, “r”);               /* reopen file */

            while((c=getc(f1))!=EOF)   /*read character from file INPUT*/
                        printf(“%c”, c);                  /* print character to
screen */
```