Input/Output and the Operating Systems

Review

- union
  - sizeof( ) function
- bit fields

I/O Functions

- Formatted I/O
  - printf( ) and scanf( )
  - fprintf( ) and fscanf( )
  - sprintf( ) and sscanf( )

```
int   fprintf(FILE *fp, const char *format, ...);
int   fscanf(FILE *fp, const char *format, ...);

int    sprintf(char *s, const char *format, ...)
int    sscanf(const char *s, const char *format,
```

- FILE structure is defined in stdio.h
  - you don’t want to know details now
- Three standard file pointers defined in stdio.h
  - stdin
  - stdout
  - stderr

I/O Functions

Declarations and initializations			
char c = 'A', s[] = "Blue moon!";			
Format	Corresponding argument	How it is printed in its field	Remarks
%c	c	"A"	Field width 1 by default
%2c	c	" A"	Field width 2, right adjusted
%-3c	c	"A "	Field width 3, left adjusted
%s	s	"Blue moon!"	Field width 10 by default
%3s	s	"Blue moon!"	More space needed
%.6s	s	"Blue m"	Precision 6
%-11.8s	s	"Blue moo "	Precision 8, left adjusted

Declarations and initialisations			
int i = 123; double x = 0.123456789;			
Format	Corresponding argument	How it is printed in its field	Remarks
%d	i	"123"	Field width 3 by default
%05d	i	"00123"	Padded with zeros
%7o	i	" 173"	Right adjusted, octal
%-9x	i	"7b "	Left adjusted, hexadecimal
%-#9x	i	"0x7b "	Left adjusted, hexadecimal
%10.5f	x	" 0.12346"	Field width 10, precision 5
%-12.5e	x	"1.23457e-01 "	Left adjusted, e-format

```
char str1[]="1 2 3 go", str2[100], tmp[100];
int  a, b, c;

sscanf(str1, "%d%d%d%s", &a, &b, &c, tmp);
sprintf(str2, "%s %s %d %d %d\n", tmp, tmp, a, b, c);
printf("%s", str2); /* will print go go 1 2 3 */
```

fopen() and fclose()

- A file should be opened before being used
  - why?
- After used, it is better to be closed
  - to flush the buffer (fflush)
- BTW, what is a file?
  - a sequence of bytes(characters)
  - these bytes can be accesses sequentially/randomly

Standard Files

- The system opens the three standard files
  - stdin, stdout, stderr
- printf/scanf functions work with stdout/stdin
  - screen/keyboard in most cases

fopen()

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

- performs housekeeping to use a file
  - access right
  - availability
  - data structures for a file
- successful call returns a file pointer
- unsuccessful call returns NULL

- mode is either “r” or “w” or “a”
  - for “w” or “a”, a new file is created if it doesn’t exist
- “r+” for open a text file for read/write
- “rb” to read a binary file
- fseek()
  - initially, fp points the beginning of the file
  - the fp points the next byte to be accesses
  - fseek() sets the value of fp

```
/* Replicate a file with caps. */

#include <ctype.h>
#include <stdio.h>
#include <stdlib.h>

FILE   *gfopen(char *filename, char *mode);

int main(int argc, char **argv)
{
   int    c;
   FILE   *fp, *tmp_fp;

   if (argc != 2) {
      fprintf(stderr, "\n%s%s%s\n\n%s\n\n",
       "Usage:  ", argv[0], "  filename",
       "The file will be doubled and some
    letters capitalized.");
      exit(1);
   }
   fp = gfopen(argv[1], "r+");
   tmp_fp = tmpfile();
   while ((c = getc(fp)) != EOF)
      putc(toupper(c), tmp_fp);
   rewind(tmp_fp);
   fprintf(fp, "---\n");
   while ((c = getc(tmp_fp)) != EOF)
      putc(c, fp);
   return 0;
}

FILE *gfopen(char *filename, char *mode)
{
   FILE   *fp;

   if ((fp = fopen(filename, mode)) == NULL)
    {
      fprintf(stderr, "Cannot open %s -
    bye!\n", filename);
      exit(1);
   }
```

tmpfile( ) creates a temporary file that will be deleted when it is closed or when the program exits

```c
/* Write a file backwards. */

#include <stdio.h>

#define   MAXSTRING   100

int main(void)
{
   char   fname[MAXSTRING];
   int    c;
   FILE   *ifp;

   fprintf(stderr, "\nInput a filename:  ");
   scanf("%s", fname);
   ifp = fopen(fname, "rb");        /* binary mode for MS_DOS */
   fseek(ifp, 0, SEEK_END);         /* move to end of the file */
   fseek(ifp, -1, SEEK_CUR);        /* back up one character */
   while (ftell(ifp) > 0) {
      c = getc(ifp);                /* move ahead one character */
      putchar(c);
      fseek(ifp, -2, SEEK_CUR);     /* back up two characters */
   }
   return 0;
}
```

`fseek(file_ptr, offs`

the file position indic

that represents `offset`

Executing Commands

```
int   system(const   chat *s);
```

- system("date"); /* "date" is a command */
  - legal set of commands differ system to system

Timing

- there is a very accurate clock inside a computer
- <time.h> file defines clock_t and time_t

- clock_t clock(void);
  - the time used by this program

- time_t time(time_t *p);
  - seconds elapsed since 1/1/1970