ESc101: File Access

Instructor: Krithika Venkataramani

Semester 2, 2011-2012

File access

- A large number of inputs can be read from a file
- You had done so using

./a.out < filename.ext

- Files can be read explicitly in the program using file pointers
- Outputs can also be stored in a file
- A file is accessed using a pointer to a file

FILE *fp;

- To read or write a file, it must be first opened using fopen()
- fopen returns a file pointer
  - Two strings as parameters
  - First is the full name (path) of the file
  - Second is the mode of operation

File access

- Mode of operation
  - r for reading: error if file does not exist
  - w for writing: file created if does not exist, overwritten if exists
  - a for appending: file created if does not exist, appended at end of file if it exists

fp = fopen ("fin. txt", "r");

- After completing operations on a file, it must be closed using fclose()

fclose(fp);

File input/output

- Analogous to scanf and printf, for file operations, there are

fscanf() and fprintf()

- Parameters are same except an extra parameter in the beginning
- The first parameter is the file pointer

fscanf (fp , "%d", &n);

fprintf (fp , "%d", n);

- getc() and putc() in file operations are analogous to getchar() and putchar()
- Require the file pointer as argument

char c = getc(fp);

putc(c, fp);

Checking for File end

- To check if the end of the file has been reached, feof() is used
- The following code stores integers in an array till the end of the file

```
j = 0;
while (! feof (fp))
    fscanf (fp , "%d", &a[j++]) ;
```

- a[j++]: uses the current value of j as index of a, and then increments j
- First value will be a[0]
- a[++j]: increments j first, and then uses the value of j as index of a
- First value will be a[1]

Input file contents

```
3
4 -7 5
2 5 1
1 6 -8
First Last
19.34
char12
```

File input/output

```
# include <stdio .h>
# include <string .h>
int main ()
{
FILE *ifp , * ofp ;
int n;
int a [3][3];
char str [2][10];
double d;
char c [10];
int i, j;
int count ;
ifp = fopen ("fin. txt ", "r");
ofp = fopen (" fout .txt", "w");
```


Krithika Venkataramani (krithika@cse.iitk.ac.in)

file read

```
fscanf (ifp , "%d", &n);
for (i = 0; i < 3; i++)
    for (j = 0; j < 3; j++)
        fscanf (ifp , "%d", &a[i][j]);
for (i = 0; i < 2; i++)
    fscanf (ifp , "%s", str [i]);
fscanf (ifp , "%lf", &d);
i = 0;
while (! feof ( ifp ))
{
    fscanf (ifp , "%c", &c[i]);
    i ++;
}
```


Krithika Venkataramani (krithika@cse.iitk.ac.in)

file write

```
count = i - 1;
printf (" Number of characters read = %d\n", count );
fprintf (ofp , "%d\n", (n + 2));
for (i = 0; i < 3; i++)
{
    for (j = 0; j < 3; j++)
        fprintf (ofp , "%d\t", a[j][i]);
    fprintf (ofp , "\n");
}
for (i = 0; i < 2; i++)
    fprintf (ofp , "%s ", strcat (str [i], "."));
fprintf (ofp , "\n");
fprintf (ofp , "%lf\n", -d);
i = 0;
```


Krithika Venkataramani (krithika@cse.iitk.ac.in)

```
while (i < count )
{
    fprintf (ofp , "%c", c[i]);
    i ++;
}
fprintf (ofp , "\n");
fclose ( ifp );
fclose ( ofp );
}
```


Krithika Venkataramani (krithika@cse.iitk.ac.in)

Command line arguments

- Arguments can be passed to a program when executing it
- Command line arguments are useful for
  - Passing filenames (e.g. gcc filename)
  - Specifying a set of parameters or options
- Arguments are supplied as strings after the program name
- These are specified as parameters of the main function

int main (int argc , char * argv [])

- argc denotes the number of arguments passed
- argv is the array that stores all the arguments as strings
  - The first argument argv[0] is the program name (e.g., a.out)
- Only char * or strings can be passed
- If numbers are needed, they must be appropriately converted
  - Use atoi() and atof() functions from stdlib.h

Command line arguments

```
#include <stdio.h>
#include <stdlib.h>
int main(int argc, char *argv[])
{
    int i;
    int n;
    float f;
    double d;
    printf("The number of arguments passed is %d\n", argc);
    printf("The arguments passed are:\n");
    for (i = 0; i < argc; i++)
        printf("%s\n", argv[i]);
    n = atoi(argv[1]);
    f = atof(argv[2]);
    d = atof(argv[3]);
    printf("%d\t%f\t%lf\n", n, f, d);
}
```

File Copy

```
#include <stdio.h>
void filecopy(FILE *fp1, FILE *fp2)
{
    char c;
    while ((c = getc(fp1)) != EOF)  // when file ends, EOF is read
        putc(c, fp2);
}
int main(int argc, char *argv[])
{
    FILE *fp1, *fp2;
    if (argc != 3)  // error handling
    {
        printf("Program requires two file arguments\n");
        return -1;     // if error, terminate
    }
```


File Copy (cont.)

```
    fp1 = fopen(argv[1], "r");     // directly using
    fp2 = fopen(argv[2], "w");     // arguments as filenames

    filecopy(fp1, fp2);   //use function defined to copy files

    fclose(fp1);
    fclose(fp2);
}
```

The content of most of these slides are from the lecture slides of Prof. Arnab Bhattacharya

Krithika Venkataramani (krithika@cse.iitk.ac.in)