PH502: Scientific Programming Concepts

Irish Centre for High End Computing (ICHEC)

September 23, 2020

Overview

- Next we we see how to read and write from a file.

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

- Opens the system file called *filename* and returns a *stream pointer* to it, if successful. Returns NULL otherwise.
- The *mode* parameter defines which permissions are required on the file. The most common you'll need are "r" (read), "w" (write) and "a" (append).
- When done with a file, close it with `fclose()`.

```
FILE *f;
f = fopen("/home/bruno/project/readme.txt", "w");
// [...]
// write some content into the file
// [...]
fclose(f);
```

Read/Writing to File

FORTRAN, OPEN

- Read/writing to the screen and to file are very similar.
- However, first the file must be opened, on a unit other than 5 or 6. The unit number must be a positive integer less than $2^{31} - 1$.
- The open statement has a set of qualifiers, some are the same as the write statement.

```
open(UNIT=n,FILE=name,ACCESS=s1,ACTION=s2,FORM=s3,&
     STATUS=s4,IOSTAT=ivar,ERR=label)
```

Qualifier	Description
FILE	name of file
ACCESS	'sequential' or 'direct'
ACTION	'write', 'read' or 'readwrite'
FORM	'formatted' or 'unformatted'
STATUS	'old', 'new', 'scratch'
	'replace' or 'unknown'

Qualifiers

- In FORTRAN, human or machine readable files can be read or written. We shall concern ourselves with human readable files.
- Thus "ACCESS='sequential'" and "FORM='formatted'".
- "STATUS" relates to the existence of the file. For example "STATUS='old'" means that the file should already exist, if not then an error is generated. Use "unknown" if the status of the file is variable.
- Below is an example of opening and closing a human readable file.

```
   open(unit=1,file='myfile.dat',status='old',&
        form='formatted',access='sequential',err=10)

   read(1,*) a,b,c
10 continue
   close(unit=1,status='keep')
```

int fprintf(FILE* stream, const char* format, ...)

- Generic version of `printf()` that writes to any given *stream* instead of standard output.

```c
FILE *f;
int value=42;

f = fopen("readme.txt", "w");
fprintf(f, "%d", value);
fclose(f);
```

fscanf(FILE* stream, const char* format, ...)

- Generic version of `scanf()` that reads from any given *stream* instead of standard input.
- Returns the number of items read or the `EOF` constant if end of file has been reached.

```
FILE *f;
int value;

f = fopen("readme.txt", "r");
if (f == NULL){
    printf("!! can't open file\n");
    exit(1);
}
fscanf(f, "%d", &value);
printf("The value in the file is %d.\n", value);

fclose(f);
```

Example: copying a file

```
int myfilecopy(char *sourcefile, char *destfile){
    char buff[1024];
    FILE *s, *d;
    int endoffile=1;
    s = fopen(sourcefile, "r");
    d = fopen(destfile, "w");
    while ( endoffile != EOF) {
      endoffile = fscanf(s, "%s\n", &buff);
      fflush(s);   /* Flush input stream */
      fprintf(d, "%s\n", buff);
    }
    fclose(s);
    fclose(d);
}
```

Read/Write to File

FORTRAN, READ&WRITE

- In FORTRAN there are no special functions to read and write to file.
- Use the "read" and "write" statements in the same way as you would to screen. The only difference being the unit number. The file must be opened before you access it.
- For human readable files, each line in the file is a separate record. By default after each read or write statement we advance to the next record.