comp10002
Foundations of Algorithms

Semester Two, 2025

Files

© The University of Melbourne, 2025
Lecture slides prepared by Alistair Moffat

Chapter 11 – Concepts

A file pointer is a connection between an executing program and a input or output device, often but not always on permanent storage.

Files need to be *opened* before they can be used; there are several different modes that can be used to open a file.

Text files can be manually edited and viewed using standard tools. Binary files provide faster input and output for arrays and structures, but cannot be processed manually.

Chapter 11 – Concepts

Three files are always provided when a program is executing:

- `stdin`, for input form the keyboard, and available for redirection by the shell
- `stdout`, for output to the screen, and available for redirection by the shell
- `stderr`, output to the screen, and available for separate redirection by the shell

In a program, `printf(..)` is a call to `fprintf(stdout,..)`; similarly `scanf(..)` just calls `fscanf(stdin,..)`.

Error messages are generated as `fprintf(stderr,"xx",yy)`.

File access modes

The function `fopen()` takes two arguments. The first is a filename, as a string. The second is an access mode, one of

- `"r"` – open for reading
- `"w"` – open for writing, previous contents deleted at moment of opening
- `"a"` – open for appending, previous contents retained

If a `"+"` is appended, the operations `fseek()` and `ftell()` are also available, for random access seek/read/rewrite processing.

Reading and writing

Functions `fread()` and `fwrite()` are used to transfer blocks of data between files and arrays, in exact internal format. No conversions of any sort are performed.

The file pointer used is of type `FILE*`, and must be opened before it is used for either operation.

Binary input and output – Standard recipe


```
type_t *tptr;
FILE *datafyle;
n = ... ;
tptr = (type_t*)malloc(n*sizeof(*tptr));
assert(tptr);
if ((datafyle = fopen(FYLENAME, "r")) == NULL) {
    fprintf(stderr, "cannot read from  %s\n", FYLENAME);
    exit(EXIT_FAILURE);
}
if (fread(tptr, sizeof(*tptr), n, datafyle) != n) {
    fprintf(stderr, "read error on %s\n", FYLENAME);
    exit(EXIT_FAILURE);
}
fclose(datafyle);
```

Binary input and output – Standard recipe


```
/* do stuff with array at *tptr, including realloc()
   if required, and adjust n if so */

if ((datafyle = fopen(FYLENAME, "w")) == NULL) {
    fprintf(stderr, "cannot write to %s\n", FYLENAME);
    exit(EXIT_FAILURE);
}
if (fwrite(tptr, sizeof(*tptr), n, datafyle) != n) {
    fprintf(stderr, "write error on %s\n", FYLENAME);
    exit(EXIT_FAILURE);
}
fclose(datafyle);
```

Chapter 11 – Program examples

- `twolines.c`
- `fread.c`
- `mergefiles.c`

Chapter 11 – Summary

Files connect transient run-time data with permanently stored data.

Functions are provided that allow reading and writing of permanent files, and for seeking to random locations within them.

When a program starts executing, it will typically read some initial data from disk. When it terminates, it might create an updated file.