PURDUE UNIVERSITY®

CS 240: Programming in C

Lecture 4: More on File I/O

Prof. Jeff Turkstra

Announcements

- Homework 2 available today!
  - Start it early, it's considerably more involved than Homework 1
- How was Homework 1?
- This class is going to begin moving faster after today
- Read Chapter 7 in K&R
  - ...and/or Chapter 13 in Beej's

Reading data in C

- C is a little different than Java when it comes to reading data
- Think printf() in reverse…

```
scanf("%s %d", buffer, &int_var);
```

- Returns number of successful conversions

Same example with a file

No error checking...

```
#include <stdio.h>

int main() {
   FILE *file_ptr = 0;
   file_ptr = fopen(“xyz”, “w”);

   fprintf(file_ptr, “Hello, world!\n”);

   fclose(file_ptr);

   return 0;
}
```

Same example with a file
Proper error checking

```
#include <stdio.h>

int main() {
  FILE *file_ptr = 0;

  file_ptr = fopen(“xyz”, “w”);
  if (file_ptr == NULL) {
    fprintf(stderr, “Can’t open.\n”);
    return 1;
  }

  fprintf(file_ptr, “Hello, world!\n”);
  fclose(file_ptr);
  file_ptr = NULL;

  return 0;
}
```

More file operations

```
int access(char *file_name, int mode);
```

- Used to check that a file can be opened with the specified mode

```
int feof(FILE *file_pointer);
```

- Used to determine if end-of-file was reached by the previous read

```
int ferror(FILE *file_pointer);
```

- Check for any error condition for the file

```
void clearerr(FILE *file_pointer);
```

access()

- Used to check if a file can be accessed before trying to open it

```
int access(char *file_name, int mode);
```

- Mode is not the same as fopen()'s mode. It is one of:
  R_OK: Check for read access
  W_OK: Check for write access
  F_OK: Check for existence
- Returns zero on success

Example of access()

```
#include <stdio.h>
#include <unistd.h> /* for access */

int main(int argc, char *argv[]) {
  if (argc != 2) {
    printf("Specify a file.\n");
    return 1;  /* indicate user error to OS */
  }

  if (access(argv[1], R_OK) == 0) {
    printf("%s is readable.\n", argv[1]);
  }
  else {
    printf("%s is not readable.\n", argv[1]);
  }

  return 0; /* indicate success to OS */
}
```

Try that on your own…

- When people ask me for ways to improve their skills, I always recommend practice
- Try that previous program on your own
- Try to implement it without looking at the lecture slide first. Then read it
- Write it down by yourself
- Type it in by yourself
- Modify it to test for writability or existence or all three conditions simultaneously

Purdue trivia

- "Harvey Washington Wiley was the first professor of chemistry, the first state chemist, the first ROTC instructor, the first baseball coach, and the 'father of the U.S. pure food and drug law.' Yet, the Purdue board of trustees once censored him for riding a bicycle - considered unseemly conduct by a faculty member."

  - A Century and Beyond,
    by Robert W. Topping

feof()

- When we're reading from a file, we need to be able to determine when we've hit the end!
- Hint: if you keep on reading after the last line, you often appear to get the last line over and over again

```
int feof(FILE *file_pointer);
```

- Note: indicates EOF (end-of-file) after the last valid read
- Returns non-zero if we have hit EOF

A program that uses feof()

```
#include <stdio.h>

int main(int argc, char *argv[]) {
  FILE *fp = NULL;
  char buf[100] = "";

  fp = fopen(argv[1], "r");

  fscanf(fp, "%[^\n]\n", buf);  /* read a line  */
  while (feof(fp) == 0) {       /* test for EOF */
    printf("%s\n", buf);        /* print line   */
    fscanf(fp, "%[^\n]\n", buf); /* read a line */
  }

  return 0;
}
```

A program that doesn't use feof()

```
#include <stdio.h>

int main(int argc, char *argv[]) {
  FILE *fp = NULL;
  char buffer[100] = "";
  int status = 0;

  fp = fopen(argv[1], "r");
  if (fp == NULL) return 0;

  while (1) {
    status = fscanf(fp, "%[^\n]\n", buffer);
    if (status == 1) printf("%s\n", buffer);
    else break;
  }

  fclose(fp);
  return 0;
}
```

Other errors…

- What if you run out of disk space while you're writing a file? How do you check for this?
- `ferror()` looks for various errors (including EOF)
- Call it after every read or write to figure out if something bad happened
- If you can correct it somehow, use `clearerr()` to clear the file pointer's internal error flag

ferror() example

```
#include <stdio.h>

int main(int argc, char *argv[]) {
  FILE *fp = NULL;

  fp = fopen(argv[1], “w”);
  do {
    fprintf(fp, “Hello, world.\n”);
  } while (ferror(fp) == 0);

  printf(“The disk is full!\n”);
  return 0;
}
```

Homework 2

- Use fscanf() inside a loop to read each line
- Look at fscanf examples in book
- Read the “printf FAQ” on the CS 240 website
- Remember: fscanf() returns the number of elements that it reads. You should use this as a test for end-of-file instead of using feof() or ferror()

fscanf() example

- If you have a chunk of code like this:

```
char buffer[100];
int val;
float cash;
int x;
x = fscanf(fp, “%s, %d, %f\n”, buffer,
                &val, &cash);
```

- Is it correct?
- What form of line are we scanning in?
- What is the return value?

%[] - set of characters

- Can specify ranges
  - %[0-9], %[A-Z], %[0-9A-z]
- Can invert
  - %[^0-9], %[^ABCDF]
- To match a], make it the first in the set
- To match a hyphen, make it the last character

- For %[] and %s, field width is important

fscanf() return values

- fscanf() will return either:
  - The number of variables it successfully scanned
  - (-1) if we've hit the end of file
- It may return a zero if there's a blank line at the end of the file
- To find the end of file, check if the return value is either 0 or -1
- To catch any other error, check if the return value is less than the number of variables to scan

What can go wrong?

```
#include <stdio.h>

int main() {
  int z = 0xffffffff;
  char buf[8];
  int y = 0xffffffff;

  scanf("%s", buf);

  printf("read: %s\n", buf);
  printf("z = %d, y = %d\n", z, y);

  return 0;
}
```

For Next Lecture

- Keep working on HW1
- (Re-)Read Chapter 7 of K&R
  - …and/or Chapter 13 in Beej's
- Understand the following functions:
  - ftell()
  - fseek()
  - fgets()
  - fputs()
  - assert()

Boiler Up!