**CS241 Lecture 5 September 2, 2015 - Angrave and Karrels - I/O**

How do I declare a stuct in C?

How do I reference a member of a struct?

What is the shortcut for dereferencing a pointer of a struct?

Should I cast the return value of malloc?

With C (FILE\*) I/O, how do I:

* Open a file for reading? For writing?
* Write formatted text?
* Read formatted text?
* Write byte arrays?
* Read byte arrays?
* Close a file?

With POSIX (file-descriptor) I/O, how do I:

* Open a file for reading? For writing?
* Write formatted text?
* Read formatted text?
* Write byte arrays?
* Read byte arrays?
* Close a file?

Which set of I/O calls are built on top of which?

Which set of I/O calls are buffered by default?

Why buffer output? Why not?

What do these other I/O routines do?

* getchar, getc, fgets, putc, fputc
* sprintf()
* gets()
* fgets()
* getline() - #define \_GNU\_SOURCE
* puts()
* fputs()
* scanf() / fscanf() / sscanf()

How do I use scanf() to:

* read an integer?
* read a string?
* skip over white space?
* skip over a string or integer?
* tell you how many characters it has consumed?

How do I read command line arguments?

How are text files different on Unix vs. Windows?

**Demo 1 - When will it crash?**

#include <stdio.h>

#include <stdlib.h>

int main() {

int \*p = malloc(sizeof(int) \* 4);

int i=0;

while (1) {

printf("write to p[%d]\n", i);

// fprintf(stderr, "write to p[%d]\n", i);

// To redirect stdout to a file:

// ./program > outfile

// To redirect stderr to a file (with the bash shell):

// ./program &> outfile

// If the output is sent to a file, why does stderr produce

// more output than stdout?

p[i] = 0;

i++;

}

free(p);

printf("No crash??\n");

return 0;

}

**Demo 2 – Demonstrate stderr and stdout**

#include <stdio.h>

#include <stdlib.h>

#include <time.h>

int main(int argc, char \*\*argv) {

if (argc != 2) {

fprintf(stderr, "\n l5b <count>\n\n");

return 1;

}

int count = atoi(argv[1]);

if (count <= 0) return 0;

srand(time(NULL));

for (int i=0; i < count; i++) {

int n = rand();

printf("%d\n", n);

if (n > 2130000000) fprintf(stderr, "Error: %d\n", n);

}

return 0;

}

**Demo 3 – command line arguments**

#include <stdio.h>

int main(int argc, char \*\*argv) {

for (int i=0; i < argc; i++) {

printf("[%d] %s\n", i, argv[i]);

}

return 0;

}

**Demo 4 – carriage return without newline**

#include <stdio.h>

#include <unistd.h>

int main() {

for (int i=0; i <= 10; i++) {

printf("\rtime remaining: %d", 10-i);

sleep(1);

}

printf("done.\n");

return 0;

}