ECE357: Problem 1 Write-Up

Camille Chow

September 24, 2017

Source Code - minicat.c

```
#include <stdio.h>
#include <fcntl.h>
#include <errno.h>
#include <string.h>
#include <getopt.h>

int buff_size, fdout, fdin, no_input, input_count;
char* out_filename;
char* in_filename;

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

    fdin = 0;
    fdout = 1;
    buff_size = 16384;
    out_filename = "stdout";
    in_filename = "stdin";
    no_input = 0;

    //process command line arguments
    int arg;

    while ((arg = getopt(argc, argv, "b:o:")) != -1) {
        //search for flags
        switch (arg) {
            //set buffer size
            case 'b':
                buff_size = atoi(optarg);
                break;
            //open output file
            case 'o':
                out_filename = optarg;

                if ((fdout = open(out_filename, O_WRONLY|O_CREAT|O_TRUNC, 0666)) < 0) {
                    fprintf(stderr, "Could not open file '%s' for writing: %s\n", out_filename,
                    strerror(errno));
                    return -1;
                }
                break;
        }
    }
```

```
//determine number of input files
if (optind >= argc) {
    input_count = 1; //if no input specified, the input "file" is stdin
    no_input = 1;
}
else {
    input_count = argc - optind;
}

//begin concatenating
char buffer[buff_size*8];
int n, m, p;
n = -1;
m = -1;
p = -1;

//reading and writing main loop
for (int i = 0; i < input_count; i++) {

    if (no_input == 1) {
        in_filename = "-";
    }
    else {
        in_filename = argv[optind+i];
    }

    if ((strcmp(in_filename, "-") == 0)) {
        fdin = 0;
        in_filename = "stdin";
    }
    //open input file
    else if ((fdin = open(in_filename, O_RDONLY)) < 0) {
        fprintf(stderr, "Error opening file '%s' for reading: %s\n", in_filename,
        strerror(errno));
        return -1;
    }

    //read to buffer
    while ((n = read(fdin, buffer, buff_size)) != 0) {
        if (n < 0) {
            fprintf(stderr, "Error reading from input file '%s': %s\n", in_filename,
            strerror(errno));
            return -1;
        }
        //write from buffer
        else {
            while (m < n) { //in case of partial writes
                if ((m = write(fdout, buffer, n)) < 0) {
                    fprintf(stderr, "Error writing to output file '%s': %s\n", out_filename,
                    strerror(errno));
                    return -1;
                }
                n -= m;
```

```
                    m = 0;
                }
            }
        }

        //close input file (if it is a file)
        if ((fdin != 0) && (p = close(fdin)) != 0) {
            fprintf(stderr, "Error closing input file '%s': %s\n", in_filename, strerror(errno));
            return -1;
        }
    }

    //close output file (if it is a file)
    if ((fdout != 1) && (p = close(fdout)) != 0) {
        fprintf(stderr, "Error closing output file '%s': %s\n", out_filename, strerror(errno));
        return -1;
    }

    return 0;
}
```

Sample Runs

```
Camilles-MacBook-Air:OS_hw1 camille$ ./minicat input1.txt input2.txt
contents of input1contents of input2
Camilles-MacBook-Air:OS_hw1 camille$
```

Figure 1: Successful run

```
Camilles-MacBook-Air:OS_hw1 camille$ ./minicat -b 2 input1.txt input2.txt input3.txt
contents of input1contents of input2
Error opening file 'input3.txt' for reading: No such file or directory
Camilles-MacBook-Air:OS_hw1 camille$
```

Figure 2: Opening error (input file did not exist)

```
Camilles-MacBook-Air:OS_hw1 camille$ ./minicat -b 2 /Volumes/UNTITLED/input.txt
Error reading from input file '/Volumes/UNTITLED/input.txt': Device not configured
Camilles-MacBook-Air:OS_hw1 camille$
```

Figure 3: Reading error (removed USB device storing input file during runtime)

```
Camilles-MacBook-Air:OS_hw1 camille$ ./minicat -b 2 /Volumes/UNTITLED/input.txt
Error reading from input file '/Volumes/UNTITLED/input.txt': Device not configured
Camilles-MacBook-Air:OS_hw1 camille$
```

Figure 4: Writing error (removed USB device storing output file during runtime)

Raw Data

Buffer Size (bytes)	Real Time	User Time	Sys Time	MB/sec (based on real time)
1	13.051	1.117	11.713	7.662E-08
2	6.449	0.555	5.797	3.101E-07
4	3.416	0.290	3.107	1.171E-06
8	1.665	0.146	1.523	4.804E-06
16	0.853	0.073	0.770	1.875E-05
32	0.436	0.036	0.377	7.345E-05
64	0.230	0.020	0.206	2.787E-04
128	0.119	0.010	0.104	1.076E-03
256	0.081	0.006	0.063	3.160E-03
512	0.048	0.004	0.040	0.011
1024	0.035	0.002	0.023	0.030
2048	0.041	0.002	0.016	0.050
4096	0.021	0.001	0.013	0.198
8192	0.025	0.001	0.008	0.328
16384	0.014	0.001	0.006	1.143
32768	0.016	0.001	0.005	2.048
65536	0.054	0.001	0.005	1.214
131072	0.012	0.001	0.004	10.627
262144	0.023	0.001	0.004	11.565

Figure 5: Raw data averaged over three runs at each buffer size (all times in seconds)

Analysis

A simple python script calling the time(1) command was used to take measurements for real, user, and system time to run the minicat program. The program was run on a MacBook Air running OS X (version 10.9.5).

The raw data averaged over three runs is presented in Table 1, and plotted in the figure below. The plot shows that increasing buffer size caused an exponential decrease in run time. It makes sense that this relationship would exist because with a larger buffer size, you can load more characters at a time. Being able to read/write more characters at a time requires fewer system calls, at the expense of more memory being used for the buffer.

Question to ponder: You can specify an input file whose name is just a single hyphen by writing the file path in front of it, or if it is in the current directory: ./- This will indicate that it is a file and not a special input or command-line flag.

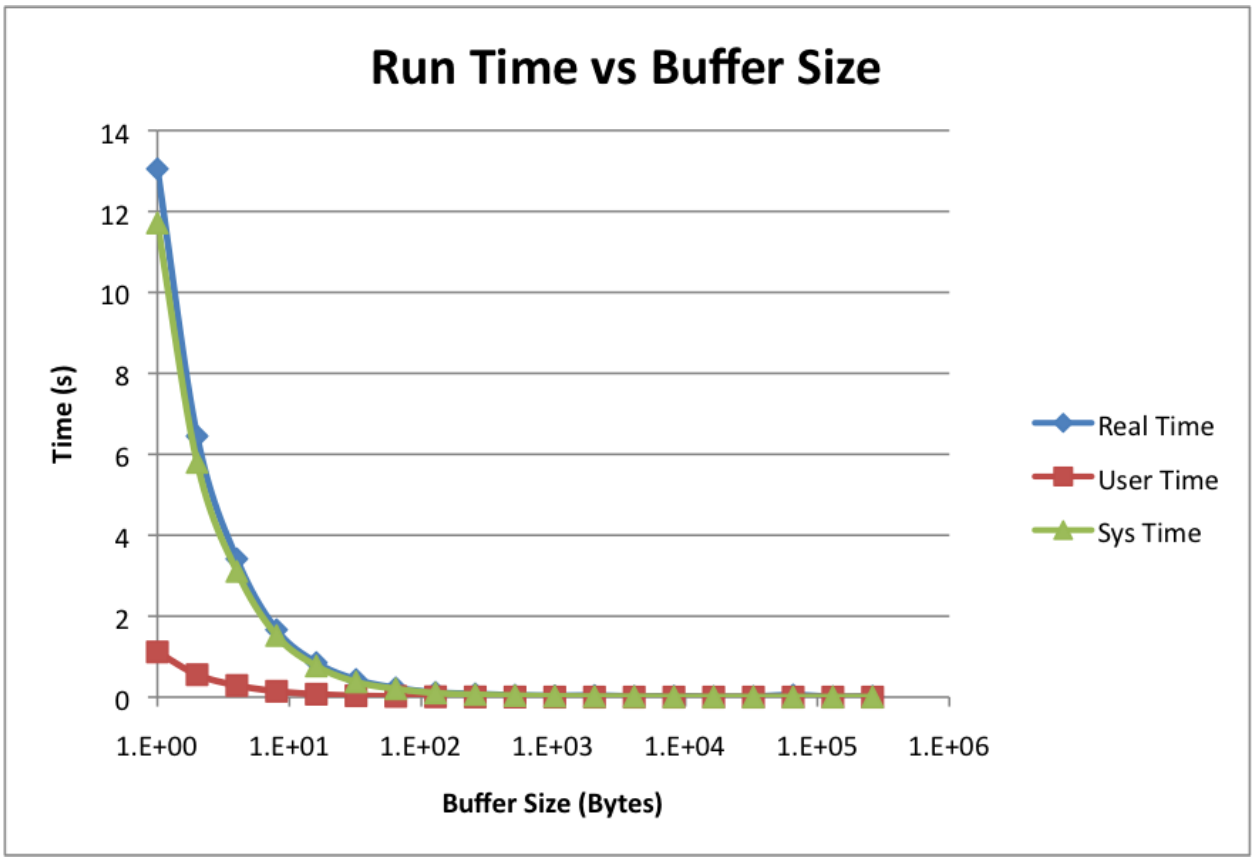

Figure 6: Plot of data from Table 1, showing exponential decay of run times as buffer size is doubled