```
// Christopher Kim
// Operating Systems, Fall 2017
// Problem Set 1

#include <stdio.h>
#include <string.h>
#include <stdlib.h>
#include <fcntl.h>
#include <errno.h>
#include <unistd.h>
#include <sys/types.h>
#include <sys/stat.h>

int openFile(char **argv, int i, int f) {
	int fd;
	fd = open(argv[i], f, 0777);
	if (fd < 0) {
		fprintf(stderr, "Error: Opening the infile %s: %s\n", argv[i], strerror(errno));
		exit(-1);
	}
	return fd;
}
void closeFile(int fd) {
	if (fd > 2) {
	// Do not want to close 0, 1 or 2 streams
		int nclose = close(fd);
		if (nclose < 0) {
			fprintf(stderr, "Error: Closing the file: %s\n", strerror(errno));
			exit(-1);
		}
	}
}
int readFile(int fd, char *input, int size) {
	int nread = read(fd, input, size);
	if (nread < 0) {
		fprintf(stderr, "Error: Reading the infile: %s\n", strerror(errno));
		exit(-1);
	}
	return nread;
}
void writeFile(int fd, char *input, int size) {
	int nwrite = write(fd, input, size);
	if (nwrite != size) {
		if (nwrite < 1) {
		// covers nwrite = 0 case as well, which is almost always an error, but errno
may not be set
			if (errno) fprintf(stderr, "Error: Writing to the outfile: %s\n",
strerror(errno));
			else fprintf(stderr, "Error: Writing to the outfile\n");
			exit(-1);
		}
		else {
	//To deal with partial writes, recursively call function to start at next up char
			writeFile(fd, input+nwrite, size-nwrite);
		}
	}
}
int main(int argc, char **argv){
	int i, j, k, m; // indices
	int size = BUFSIZ; // Default Buffer size provided is 1024
	int fdr = STDIN_FILENO; // Assigns output to stdout if no input given
	int fdw = STDOUT_FILENO; // Assigns output to stdout if no -o given
	i = 1;
```

```
	while (argc > i) {
		if (strcmp(argv[i], "-b") == 0) {
			if (argc == ++i) {
				fprintf(stderr, "Error: Parsing buffer argument.\nThe buffer size
is not given.\n");
				exit(-1);
			}
			size = atoi(argv[i]);
			if(size <= 0){
				fprintf(stderr, "Error: Setting the buffer size.\n%d is not a
valid size.\n", size);
				exit(-1);
			}
			++i;
		}
		if (strcmp(argv[i], "-o") == 0){
			m = ++i;
			if(argc == m) {
				fprintf(stderr, "Error: Parsing outfile argument.\nNo outfile is
given.\n");
				exit(-1);
			}
			fdw = openFile(argv, m, O_CREAT | O_WRONLY | O_TRUNC);
			++i;
			continue;
		}
		break;
	}
	char *input = malloc(size);
	k = i;
	do {
		j = 1;
		if ((argc > k) && strcmp(argv[k], "-")) {
			fdr = openFile(argv, k, O_RDONLY);
		}
		while (j > 0) {
			j = readFile(fdr, input, size);
			writeFile(fdw, input, j);
		}
		closeFile(fdr);
		k++;
	} while((k < argc));
	closeFile(fdw);
	free(input);
	return 0;
}
```

Christopher Kim
Operating Systems Assignment #1

Functionality Testing

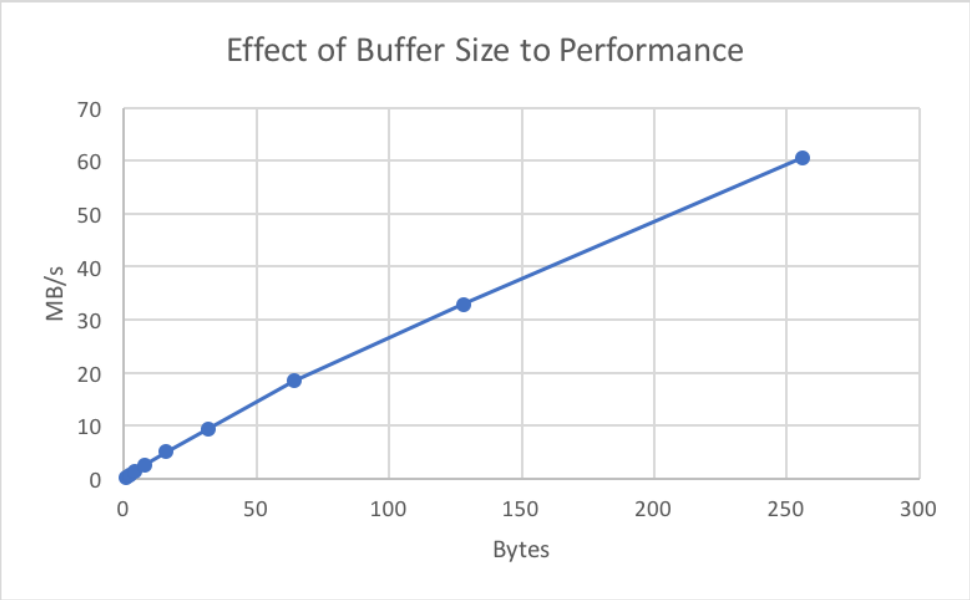
```
Christophers-MacBook-Pro-2:TEST christopherkim$ gcc -o minicat minicat.c
Christophers-MacBook-Pro-2:TEST christopherkim$ ./minicat
Hi my name is Joe
Hi my name is Joe
Christophers-MacBook-Pro-2:TEST christopherkim$ ./minicat -
Hello Joe
Hello Joe
Christophers-MacBook-Pro-2:TEST christopherkim$ ./minicat Round.txt
The wheels on the bus go round and round
round and round
hello my name is Joe.
Christophers-MacBook-Pro-2:TEST christopherkim$ ./minicat -o Tester.txt Test.txt
Christophers-MacBook-Pro-2:TEST christopherkim$ ./minicat -b 512 -o Tester2.txt Round.txt
Christophers-MacBook-Pro-2:TEST christopherkim$ ls
Round.txt       Tester2.txt     minicat.c
Tester.txt      minicat         test.txt
```

Error Testing (Using the USB Method)

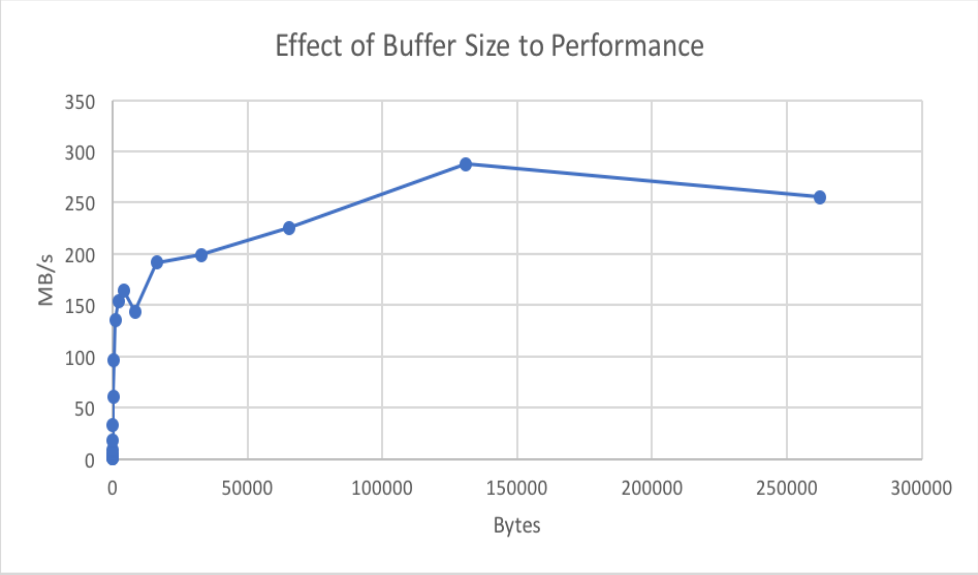
```
Christophers-MacBook-Pro-2:TEST christopherkim$ ./minicat nothere.txt
Error: Opening the infile nothere.txt: No such file or directory
Christophers-MacBook-Pro-2:TEST christopherkim$ ./minicat -b 2 -o Tester3.txt Test.txt
Error: Writing to the outfile: Device not configured
Christophers-MacBook-Pro-2:TEST christopherkim$ ./minicat -b 2 -o Testing.txt Test.txt
Error: Writing to the outfile: Input/output error
```

Bytes	(MB/s)
1	0.336
2	0.671
4	1.333
8	2.611
16	5.077
32	9.426
64	18.435
128	32.861
256	60.526
512	95.833
1024	135.294
2048	153.333
4096	164.285
8192	143.751
16384	191.667
32768	199.245
65536	225.491
131072	287.573
262144	255.455

From 1 to 256



From 1 to 256K



Three trials were conducted for each buffer size, and the average of the three were taken. As the size of the buffer is increased by a power of 2, the throughput also increased by close to a power

of 2. This is not surprising, as a doubling the buffer size halves the amount of reading and writing the program must do. To a certain point, the graph represents a logarithmic growth in base 2. However, further increasing the buffer size led to a much slower increase in throughput until it stagnated completely, as the limitations of the hardware I am using affected the doubling trend. The hardware I am using is a Macbook Pro (Retina, 13-inch, Early 2013), Processor 2.6 GHz Intel Core i5, 8 GB 1600 MHz DDR3 Memory.