CS 342 PROJECT 1 REPORT

Sıla İnci

21602415

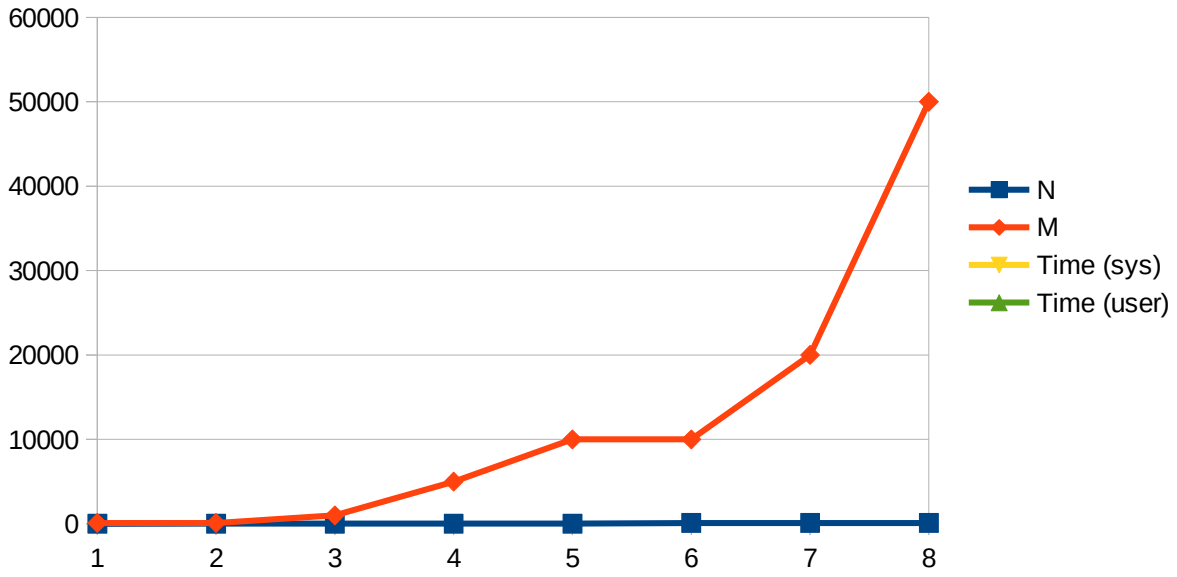
02.03.2019

- **EXPERIMENTS**

Experiment #	N	M	Time (sys)	Time (user)
1	5	100	0m0,003s	0m0,001s
2	10	100	0m0,001s	0m0,000s
3	10	1000	0m0,004s	0m0,001s
4	10	5000	0m0,005s	0m0,002s
5	10	10000	0m0,006s	0m0,002s
6	100	10000	0m0,001s	0m0,000s
7	100	20000	0m0,001s	0m0,000s
8	100	50000	0m0,001s	0m0,000s

Fig. 1: Table of Experimental Results with Time Command

Graph 1: Time vs Read/Write Operations



N	command	read-call-count	char-count
3	ps aux | sort	12596	37788
10	ps aux | sort	3787	37870
30	ps aux | sort	1263	37890

Fig. 2: Table of Experimental Results of “read-call-count” and “char-count” with Compound Commands

- **INTERPRETATIONS**

In this project, we were given the task to experiment with creating compound commands and how to connect their communication using pipes. The communication between these structures is supplied by reading and writing data from a file. When a structure or a command completes the task, it simply writes the output to that specified file and when it is time for the other command to do the task it reads from the file which has the data in it form the first process.

In this project, we have a N variable which is used in every compound command. The N is the amount of bytes we want to read or write each time we call the write() or read() syscall. This way, if the same task is given with N =5 and N = 100 the read/write operation will take more time with the variable N=5. Because we specify to the program that we want only 5 bytes to be read/written. In figure.1, we see the difference in time when the "consumer M | producer M" command is called with different M and N values. In the 5th and 6th we can examine the affect N has on the read/write operation. When the value N is 10 and the M is 10000 it takes 0m0,006s (sys) and when the value N is 100 while the M stays the same, now time is 0m0,001s (sys) because the number of time we call read/write has less than the first time.

- **CONCLUSIONS**

In conclusion, the transfer of data through pipes can be manipulated with the help of read/write syscalls. The third parameter of these syscalls, determine the number of bytes the function will read/write in each call. When we give the N to that parameter, the bigger the value N is the less read/write operation will be called because the task will be completed faster. The time also changes corresponding to the changes in the value N with this logic.

- **THE CODE**

I have written producer and consumer commands as builtins of my shell similar to "cd" and "exit".

```
void consumer_command(char **params, int val);
void producer_command(char **params, int val);

char *bilshell_commands[]={
        "consumer",
        "producer",
        "time"
};

int bil_builtins() {
  return sizeof(bilshell_commands) / sizeof(char *);
}

void consumer_command(char **params, int val)
```

```
{
	int M = atoi(params[1]);
	//printf("girdi");
	char buff[M];
	if (params[1] == NULL) {
		fprintf(stderr, "Consumer command failed\n");
	} else{
		read(0,buff,val);
	}
	//int index = 0;
}

void producer_command(char **params , int val)
{
	int M = atoi(params[1]);
	char buff[M];
	//char* random = malloc( sizeof(char) * ( M + 1 ) );
	//printf("abc");
	if (params[1] == NULL) {
		fprintf(stderr, "Consumer command failed\n");
	}
	else{
		//printf("abc");
		srandom(time(NULL));
		char alphanumerics[] =
"ABCDEFGHIJKLMNOPQRSTUVWXYZabcdefghijklmnopqrstuvwxyz1234567890";
		for( int i = 0; i < M; ++i)
		{
			//srandom(time(NULL));

			buff[i] = alphanumerics[random() % (sizeof(alphanumerics) - 1)];
		}

	}
	int m_val = M;
	while( m_val > 0)
	{

		write(1, buff,val);
		m_val -= val;
	}
	//printf("\n");
	//return random;
}

void (*bilshell_func[]) (char ** , int) = {
  &consumer_command,
  &producer_command,
  &time_command
};
```