University of Southern
Maine

**M. FENTON LIBBY**
COS 350
4 MAY 2021

# Program 6 Write-Up

Systems Programming

## *Summa*

This project was done on Honeybee. Listing for "lc1.c":

```
#include <stdio.h>
#include <string.h>
#include <fcntl.h>
#include <unistd.h>
#include <stdlib.h>

/*HEADER*/
static long size(int fd);


/*GLOBALS*/
#define EOL '\n'


/*MAIN*/
int main ( int argc , char* argv[] ) {

        /*INITIALIZATION*/
        if ( argc < 2 ){// args?
                printf("No file input, exiting...");
                exit(0);
        }

        int fd;
        long fSize;
        long lc;
        long total = 0;
        for ( int fc = 1 ; fc < argc ; fc++) {

                if ( ( fd = open ( argv [ fc ] , O_RDONLY ) ) == -1 ) {//file opens?
                        fprintf ( stderr , "Failed to open file %s" , argv [ fc ] );
                        exit(1);
                }

                fSize = size ( fd );//get file size for buffer calloc

                char* buff;
                buff = calloc ( 1, fSize + 1 );//give buffer the size of input file

                if ( ( read ( fd , buff , fSize ) ) == -1 ) {
                        close ( fd );
                        free ( buff );
                        fprintf ( stderr , "A critical failure occured when reading in the
                            file %s.\n" , argv[fc] );
                        exit ( 1 );
                }
                close ( fd );

                /*GENERAL*/
                lc = 0;
```

```
                for ( int i = 0; i < fSize; i++ ) //count EOL
                        if ( buff[i] == EOL )
                                lc++;

                //fprintf ( stdout , "The file %s has %ld lines.\n" , argv [ fc ] , lc );
                //printf ( "HELLO!\n" );
                /*CLEANUP*/
                printf ( "%ld\t%s\n" , lc , argv [ fc ] );

                free ( buff );
                total += lc;
        }
        if ( argc > 2 ) {
                printf ( "%ld\n" , total );
        }
        return 0;
}
//------------------EOM-------------------

/*AUXILLIARY METHODS*/

static long size ( int fd ) {

  long fsize;


  fsize = lseek( fd , 0L , SEEK_END );
  lseek( fd , 0L , SEEK_SET );

  return fsize;

}
```

---

Listing for "lc2.c":

```c
#include <stdio.h>
#include <string.h>
#include <fcntl.h>
#include <unistd.h>
#include <stdlib.h>
#include <sys/wait.h>
#include <sys/types.h>

/*HEADER*/



/*GLOBALS*/
int total = 0;


/*MAIN*/
int main ( int argc , char* argv[] ) {

	/*INITIALIZATION*/
	if ( argc < 2 ) {// args?
		printf ( "No files input, exiting..." );
		exit ( 0 );
	}

	int pipefd[ 2 ];
	pid_t cpid[ argc - 1 ];


	if ( pipe ( pipefd ) == -1 ) {
		perror ( "pipe" );
		exit ( 1 );
	}


	for ( int i = 0 ; i < argc - 1 ; i++ ) {

		if ( ( cpid [ i ] = fork ( ) ) == -1 ) { //create fork
			perror ( "fork" );
			exit ( 1 );
		}

		if ( cpid [ i ] == 0 ) { /* Child process control */
			dup2 ( pipefd [ 1 ] , 1 );
			close ( pipefd [ 0 ] );
			close ( pipefd [ 1 ] );

			if ( execlp ( "lc1" , "lc1" , argv [ i + 1 ] , NULL ) == -1 )
				perror("execution failed");

			exit(EXIT_SUCCESS);

		}
	}

	/* Parent Process control */
	int children = 0;
```

```
        while ( children < ( argc-1 ) ) {
                wait ( 0 );
                children++;
        }

        char buf[BUFSIZ];
        int len;
        len += read(pipefd[0], &buf[len], BUFSIZ);


        int count = 0;

        close(pipefd[0]);
        close(pipefd[1]);

        if ( ( write ( 1 , buf , len ) ) == -1 ) {
                fprintf ( stderr , "Write error" );
                exit ( 1 );
        }

        fflush ( stdout );

        return 0;
}
```

Listing for "lc2.c":

```
#include <stdio.h>
#include <stdlib.h>
#include <unistd.h>
#include <sys/wait.h>
#include <string.h>
#include <pthread.h>
#include <ctype.h>
#include <sys/types.h>
#include <fcntl.h>

void *lc1(void *);

void main(int argc, char* argv[]){


        int count = 0;
        pthread_t t[ argc - 1 ];
        int* ret[ argc - 1 ];


        for ( int i = 1 ; i < argc ; i++ ) {

                pthread_create ( (&t[i]) , NULL , lc1 , (void *) argv [ i ] );

        }

        int filesProcessed = 0;


        for ( int i = 1 ; i < argc ; i++ ) {

                pthread_join ( (t[i]), ( void** ) &( ret [ i ] ) );
                printf ( "%d\t%s\n" , *(ret [ i ]) , argv [ i ] );
                count += *(ret [ i ]);

        }

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

}



void *lc1(void *file){
        char *fileName = ( char* ) file;
        int fd , bite;
        char c[ 1024 ];
        int *lc = malloc ( sizeof ( int ) );
        *lc = 0;

        fd = open ( fileName , O_RDONLY );

        while ( ( bite = read ( fd , c , 1024 ) ) != 0 ) {
                for ( int i = 0 ; i < bite ; i++ ) {
                        if( c [ i ] == '\n' ){
                                ( *lc ) ++;
                        }
```

```
                }
        }
        close(fd);
        return lc;
}
```

Makefile:

```
prog6:lc3 lc2 lc1

lc3: lc3.c
   gcc -O2 lc3.c -pthread -o lc3

lc2: lc2.c
   gcc -O2 lc2.c -o lc2

lc1: lc1.c
   gcc -O2 lc1.c -o lc1
```

A log of the required testing results:

```
Script started on 2021-05-04 14:22:18-0400
mflibby@honeybee:~/cos350/prog/prog6$ time lc1 f*
175641 f1
26175  f10
345130 f2
65336  f3
211035 f4
53835  f5
33055  f6
13426  f7
16271  f8
54544  f9
994448

real    0m0.052s
user    0m0.025s
sys     0m0.016s
mflibby@honeybee:~/cos350/prog/prog6$ time lc2 f*
26175  f10
33055  f6
345130 f2
53835  f5
175641 f1
13426  f7
16271  f8
54544  f9
65336  f3
211035 f4

real    0m0.024s
user    0m0.022s
sys     0m0.038s
mflibby@honeybee:~/cos350/prog/prog6$ time lc3 f*
175641 f1
26175  f10
345130 f2
65336  f3
211035 f4
53835  f5
33055  f6
13426  f7
16271  f8
54544  f9
994448 Total

real    0m0.030s
user    0m0.059s
sys     0m0.025s
mflibby@honeybee:~/cos350/prog/prog6$ exit
exit

Script done on 2021-05-04 14:22:45-0400
```

## DISCUSSION

The results of the 3 run trial of each program (all results in seconds) performed on **Honeybee**:

| Timing Results | Real | User | Sys |
|---|---|---|---|
| lc1 | 0.032 | 0.017 | 0.012 |
| lc2 | 0.017 | 0.023 | 0.017 |
| lc3 | 0.027 | 0.044 | 0.013 |

It would appear that the "lc2" program performs the best. We can be confident that "lc1" wouldn't be the fastest, but the fact that it performed better than the threaded version is strange, since it simply sequenced through all of the files. It is likely that, had we taken an average over many trials, we would find that the threaded one would be the fastest, lc2 would be second, and lc1 would be third by a decent margin.