University of Southern
Maine

M. FENTON LIBBY
COS 350
4 MAY 2021

Program 4 Write-Up

Systems Programming

Summa

On Honeybee, the following code was written:

```
#include <stdio.h>
#include <stdlib.h>
#include <sys/signal.h>
#include <sys/ioctl.h>
#include <termios.h>
#include <string.h>
#include <unistd.h>
#include <sys/time.h>
#include <fcntl.h>

static long size(FILE* fp);
int println(char* buff, int lineNum, int cols);
void timerHandler(int param);
int printscrn();
void rvClear();
void rvPrint();
void rvUpdate(){
  rvClear();
  rvPrint();
}
void rvEOF();

/*----------------------------------------*/
char* buff; //file buffer
long fSize;
int count;//number of chars output from buffer
struct winsize wSize;//wSize has .ws_row and .ws_col, both measures in chars
int mode = 0;
float lag = 0.0;

#define NEXT_SCREEN 32 //next line char (32 = space)
#define SCROLL_TOG '\n' //scroll toggle char
#define SPEED_UP 'f' //speed up char
#define SPEED_DN 's' //speed down char
#define QUIT 'q' //quit char
#define LAG 2.0  //base scroll timer in seconds

#define PAGE_MODE 0
#define SCROLL_MODE 1
#define PAGE_STRING "Mode: Paging (press 'Space' to flip to the next page, or 'Enter' to
    change modes)"
/*----------------------------------------*/



int main(int argc, char** argv){
  //ensure only 1 arg was passed:
  if(argc > 2){
    fprintf(stderr, "scroll only takes one argument, the name of a file to scroll
```

```
      through, you passed %d args\n", argc);
  exit(1);
}
FILE* fp;
//open the file, exit(1) if it fails
if( ( fp = fopen(argv[1],"r") ) == NULL ){
  fprintf( stderr, "The file %s couldn't be opened, perhaps it doesn't exist...\n",
      argv[1] );
  exit(1);
}

/*file buffering*/
fSize = size(fp); //file size

buff = calloc( 1, fSize+1 ); //allocate memory for the buffer

if ( ( fread(buff, fSize, 1, fp) ) != 1){
  fclose(fp);
  fprintf(stderr, "A critical failure occured when reading in the file.\n");
  exit(1);
}

/*get/set terminal attributes*/

struct termios tinfo;
ioctl(1, TIOCGWINSZ, &wSize); //fill wSize
tcgetattr(0, &tinfo);
tinfo.c_lflag &= ~ECHO & ~ICANON;
tcsetattr(0, TCSANOW, &tinfo);


/*Setup Timer*/
signal(SIGALRM, timerHandler);


struct itimerval timer;
timer.it_value.tv_sec = 0;
timer.it_value.tv_usec = 0;
count += printscrn();
setitimer(ITIMER_REAL, &timer, NULL);
/*Handle Scrolling*/
char c = '\0';
int done = 0;
printf("\033[7m");
printf("%s", PAGE_STRING);
printf("\033[0m");
while(!done){
  //pause();
  if ( count >= fSize - 1 ) {
      timer.it_value.tv_sec = 0;
      timer.it_value.tv_usec = 0;
        timer.it_interval.tv_sec = 0;
        timer.it_interval.tv_usec = 0;
      setitimer(ITIMER_REAL, &timer, NULL);
    rvClear();
    rvEOF();
    c = getchar();
  } else {
    c = getchar();
```

```
  }
  // if (count >= fSize){
  //   done = 1;
  //   break;
  // }

  switch(c){
  case QUIT:
    timer.it_value.tv_sec = 0;
    timer.it_value.tv_usec = 0;
    setitimer(ITIMER_REAL, &timer, NULL);
    tinfo.c_lflag |= ECHO | ICANON;
    tcsetattr(0, TCSANOW, &tinfo);
    free(buff);
    fclose(fp);
    return 0;
  case SPEED_UP:
    lag /= 1.2;
    timer.it_interval.tv_sec = (int) lag;
    timer.it_interval.tv_usec = (int) ((lag - (int) lag)*1000000);
    rvUpdate();
    setitimer(ITIMER_REAL, &timer, NULL);
    break;
  case SPEED_DN:
    lag *= 1.2;
    timer.it_interval.tv_sec = (int) lag;
    timer.it_interval.tv_usec = (int) ((lag - (int) lag)*1000000);
    rvUpdate();
    setitimer(ITIMER_REAL, &timer, NULL);
    break;
  case SCROLL_TOG:
    mode = ! mode;
    rvClear();
    if (lag != 0.0){
        lag = 0.0;
      timer.it_value.tv_sec = 0;
      timer.it_value.tv_usec = 0;
        timer.it_interval.tv_sec = 0;
        timer.it_interval.tv_usec = 0;
    } else {
        lag = LAG;
      timer.it_value.tv_sec = 2;
        timer.it_value.tv_usec = 0;
        timer.it_interval.tv_sec = (int) lag;
        timer.it_interval.tv_usec = (int) ((lag - (int) lag)*1000000);
    }
    rvPrint();
    setitimer(ITIMER_REAL, &timer, NULL);
    break;
  case NEXT_SCREEN:
    rvClear();
    count += printscrn();
    rvPrint();
    break;
  default:
    continue;
  }
}
printf ( "\n" );
```

```
  /*Done, Clean Up:*/
  tinfo.c_lflag |= ECHO | ICANON;
  tcsetattr(0, TCSANOW, &tinfo);

  free(buff);
  fclose(fp);

  return 0;
}



static long size(FILE* fp){

  long fsize;

  fseek( fp, 0L, SEEK_END );
  fsize = ftell( fp );
  rewind( fp );

  return fsize;

}


int println(char* buff, int lineNum, int cols){
  int i = 0;
  for(i = lineNum; i < lineNum + cols; i++){
    putc(buff[i], stdout);
    if (buff[i] == '\n'){
      return (i+1)-lineNum;
    }
  }
  //putc('n', stdout);
  return cols;
}

int printscrn(){
  int pchars = 0;
  for(int i = 0; i < wSize.ws_row-1; i++){
    if (pchars + count == fSize){
      return pchars;
    }
    pchars += println(buff, count+pchars, wSize.ws_col);
  }
  return pchars;
}

void rvClear(){
  printf( "\033[7m" );
  for ( int i = 0; i < strlen ( PAGE_STRING ) + ( mode ? 36 : 0 ); i++ ){
     putchar ( '\b' );
  }
  printf("\033[0m");
  printf ( "\033[0K" );

}

void rvPrint(){
```

```
  printf("\033[7m");
  printf ("Mode: %s (press 'Space' to flip to the next page, or 'Enter' to change modes)"
      , mode ? "Scroll" : "Paging" );
  if ( mode ) {
      printf ( ", Current Scrolling Lag Factor: %.2f" , lag );
  }
  printf("\033[0m");
}

void rvEOF(){
    printf("\033[7m");
    printf ("End of File Reached, press Q to quit!");
    printf("\033[0m");
    fflush(stdout);
}

void timerHandler ( int param ) {
  rvClear();
  putchar('\0');
  if (count < fSize){
      count += println(buff, count, wSize.ws_col);
      fflush(stdout);
      rvPrint();
  } else {
    rvClear();
    rvEOF();
    fflush ( stdin );
  }

  fflush(stdout);
}
```

Makefile:

```
build:
   gcc scroll.c -o scroll
```

DISCUSSION

This program works for the most part, but occasionally walks off the end of the file (though I'm not sure why, I thought I had handled EOF properly but I haven't had enough to fully debug, it is likely a simple fix - once the error is actually found!)

My main issue with the program is that it freezes while the lag factor is being changed (i.e. if you hit the 'f' or 's' keys, it will stop for a moment before adjusting itself according; if you hold down one of these keys, the program will freeze until the key is released, though the lag factor will change and update in real time).