

## Benjamin Kaplan - Problem Set #5

## Sample Output

```

=====
binyamin@BenjaminButtox:~/Documents$ gcc -o problem1 problem1.c
binyamin@BenjaminButtox:~/Documents$ ./problem1
cannot handle signal: 9
cannot handle signal: 19
Signal recieved: 11
binyamin@BenjaminButtox:~/Documents$ gcc problem2.c -o problem2
binyamin@BenjaminButtox:~/Documents$ ./problem2
read buf= A
binyamin@BenjaminButtox:~/Documents$ echo $?
0
binyamin@BenjaminButtox:~/Documents$ gcc problem3.c -o problem3
binyamin@BenjaminButtox:~/Documents$ ./problem3
H
binyamin@BenjaminButtox:~/Documents$ echo $?
1
binyamin@BenjaminButtox:~/Documents$ gcc problem4.c -o problem4
binyamin@BenjaminButtox:~/Documents$ ./problem4
statbuf1 size: 5000
statbuf2 size: 5000
binyamin@BenjaminButtox:~/Documents$ echo $?
1
binyamin@BenjaminButtox:~/Documents$ gcc problem5.c -o problem5
binyamin@BenjaminButtox:~/Documents$ ./problem5
posX = 5000
lseek to: 5016
buf = B
binyamin@BenjaminButtox:~/Documents$ echo $?
0
binyamin@BenjaminButtox:~/Documents$ gcc problem6.c -o problem6
binyamin@BenjaminButtox:~/Documents$ ./problem6
cannot handle signal: 9
cannot handle signal: 19
Read 1 succeeded past end of file.
Byte read: ASCII code:0
Signal recieved: 7
binyamin@BenjaminButtox:~/Documents$ echo $?
7

```

## Problem1

```

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

int main(int argc, char ** argv){
    void sigHand(int signo){
        fprintf(stderr, "Signal recieved: %d\n", signo);
        exit(signo);
    }
    int j = 1;
    for(j = 1; j<32; j++){

```

```

    if (signal(j, sigHand) == SIG_ERR)
        fprintf(stderr, "cannot handle signal: %d\n", j);
}
int fd;
if((fd = open("testfile1.txt", O_RDWR|O_TRUNC|O_CREAT, 0666))<0){
    perror("Write error: ");
    exit(EXIT_FAILURE);
}
if(write(fd, "Hello world", 11)<0){
    perror("write error: ");
    exit(EXIT_FAILURE);
}
char *memory;
if((memory = mmap(NULL, 11, PROT_READ, MAP_SHARED, fd, 0))<0){
    perror("Mmap error: ");
    exit(EXIT_FAILURE);
}

*memory = 'A';
if(*memory == 'H'){
    exit(255);
}

return 0;
}

#include <stdio.h>
#include <string.h>
#include <errno.h>
#include <fcntl.h>
#include <stdlib.h>
#include <sys/mman.h>
int main(int argc, char ** argv){
    int fd;

    if( (fd = open("testfile.txt", O_RDWR | O_CREAT |O_TRUNC, 0666))<0){
        perror("Open error: ");
        exit(EXIT_FAILURE);
    }

    write(fd, "Hello world", 11); //This is necessary because you will
                                //get a SIGBUS if you attempt to write to the
                                //MMAPing of a file with 0 bytes size

    char *memory = malloc(10);
    if((memory = mmap(NULL, 11, PROT_READ | PROT_WRITE, MAP_SHARED, fd, 0))<0)
        perror("MMAP error: ");
        exit(EXIT_FAILURE);
    }
    char letter = 'A';
    *memory = 'A';

    int g;
    if((g = lseek(fd, 0, SEEK_SET))<0){
        perror("Seek error: ");
        exit (EXIT_FAILURE);
    }

    char* buf;
    if((buf = malloc(10)) <0){
        perror("Malloc error: ");
        exit(EXIT_FAILURE);
    }
}

```

```

    if(read(fd,buf, 1) <0){
        perror("Read error: ");
        exit(EXIT_FAILURE);
    }
    fprintf(stderr, "read buf= %s\n", buf);
    if(close(fd)<0){
        perror("closing error: ");
        exit(EXIT_FAILURE);
    }
    if(!strcmp("A", buf))
        return 0;
    return 1;
}

```

### Problem 3

=====

```

#include <stdio.h>
#include <string.h>
#include <errno.h>
#include <fcntl.h>
#include <stdlib.h>
#include <sys/mman.h>
int main(int argc, char ** argv){
    int fd;

    if( (fd = open("testfile.txt", O_RDWR | O_CREAT | O_TRUNC, 0666))<0){
        perror("Open error: ");
        exit(EXIT_FAILURE);
    }

    write(fd, "Hello world", 5); //This is necessary because you will
                                //get a SIGBUS if you attempt to write to the
                                //MMAPing of a file with 0 bytes size

    char *memory = malloc(10);
    if((memory = mmap(NULL, 4096, PROT_READ | PROT_WRITE, MAP_PRIVATE, fd, 0))<0){
        perror("MMAP error: ");
        exit(EXIT_FAILURE);
    }
    char letter = 'A';

    *memory = 'A';

    int g;
    if((g = lseek(fd, 0, SEEK_SET))<0){
        perror("Seek error: ");
        exit (EXIT_FAILURE);
    }

    char* buf;
    if((buf = malloc(10)) <0){
        perror("Malloc error: ");
        exit(EXIT_FAILURE);
    }
    if(read(fd,buf, 1) <0){
        perror("Read error: ");
        exit(EXIT_FAILURE);
    }
}

```

```

fprintf(stderr,"%s\n", buf);
if(close(fd)<0){
    perror("closing error: ");
    exit(EXIT_FAILURE);
}
if(!strcmp("A", buf))
    return 0;
return 1;
}

```

#### Problem 4

```

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

int main(int argc, char **argv){
    int fd;
    if((fd = open("testfile.txt", O_RDWR|O_CREAT|O_TRUNC, 0666))<0){
        perror("Write error: ");
        exit(EXIT_FAILURE);
    }
    int j= 0;
    char letter = 'A';
    for(j=0; j< 5000; j++){
        if(write(fd, &letter, 1)<0){
            perror("Write error: ");
            exit(EXIT_FAILURE);
        }
    }
    struct stat statbuf1;
    if(fstat(fd, &statbuf1)<0){
        perror("Stat error1: ");
        exit(EXIT_FAILURE);
    }

    if(lseek(fd, 0, SEEK_END)<0){
        perror("MMAP error: ");
        exit(EXIT_FAILURE);
    }
    fprintf(stderr,"statbuf1 size: %d\n", (int)statbuf1.st_size);
    char * memory;
    if((memory = mmap(NULL, 1, PROT_WRITE| PROT_READ, MAP_SHARED, fd,0 ))<0){
        perror("MMAP error: ");
        exit(EXIT_FAILURE);
    }
    *(memory+5001) = 'A';

    struct stat statbuf2;
    if(fstat(fd, &statbuf2)<0){
        perror("Stat error2: ");
        exit(EXIT_FAILURE);
    }
    fprintf(stderr,"statbuf2 size: %d\n", (int) statbuf2.st_size);
    int size1 = (int) statbuf1.st_size;
    int size2 = (int) statbuf2.st_size;
    if(size1 == size2)

```

```

    return 1;
    return 0;
}

```

# Problem 5

```

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

int main(int argc, char **argv){
    int fd;
    if((fd = open("testfile.txt", O_RDWR|O_CREAT|O_TRUNC, 0666))<0){
        perror("Write error: ");
        exit(EXIT_FAILURE);
    }

    int j= 0;
    char letterA = 'A';
    for(j=0; j< 5000; j++){
        if(write(fd, &letterA, 1)<0){
            perror("Write error: ");
            exit(EXIT_FAILURE);
        }
    }
    if(lseek(fd, 0, SEEK_END)<0){ // lseeks to end of file
        perror("Lseek error: ");
        exit(EXIT_FAILURE);
    }
    char *memory;
    if(!(memory = mmap(NULL, 8192, PROT_READ|PROT_WRITE,MAP_SHARED,fd,0))){
        perror("MMAP error: ");
        exit(EXIT_FAILURE);
    }
    char letterB = 'B';
    int posX;
    *(memory+5000) = 'B';

    if((posX = lseek(fd,0,SEEK_CUR))<0){
        perror("Lseek error: ");
        exit(EXIT_FAILURE);
    }
    fprintf(stderr, "posX = %d\n", posX);
    int g;
    if((g =lseek(fd, 16, SEEK_END))<0){
        perror("Lseek error: ");
        exit(EXIT_FAILURE);
    }
    fprintf(stderr,"lseek to: %d\n", g);
    char letterC = 'C';
    if(write(fd, &letterC, 1)<0){
        perror("Write error: ");
        exit(EXIT_FAILURE);
    }
    char *buf;
    if(lseek(fd, posX, SEEK_SET)<0){
        perror("lseek error:");
        exit(EXIT_FAILURE);
    }

```

```

}
if((buf = malloc(10))<0){
    perror("malloc:");
    exit(EXIT_FAILURE);
}
if(read(fd, buf, 1)<0){
    perror("read:");
    exit(EXIT_FAILURE);
}
fprintf(stderr,"buf = %s\n", buf);
if(!strcmp(buf,"B"))
    return 0;
return 1;
}

```

### Problem 6

```

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

int main(int argc, char **argv){
    void sigHand(int signo){
        fprintf(stderr, "Signal recieved: %d\n", signo);
        exit(signo);
    }

    int j = 0;
    for(j = 1; j<32; j++){
        if (signal(j, sigHand) == SIG_ERR)
            fprintf(stderr,"cannot handle signal: %d\n", j);
    }

    int fd;
    if((fd = open("testfile6.txt", O_RDWR|O_TRUNC|O_CREAT, 0666))<0){
        perror("Write error: ");
        exit(EXIT_FAILURE);
    }
    int k = 0;
    char letterA = 'A';
    for(k=0; k<1000; k++){
        if(write(fd, &letterA, 1)<0){
            perror("Write error: ");
            exit(EXIT_FAILURE);
        }
    }

    char * memory;
    if((memory = mmap(NULL, 8192, PROT_WRITE| PROT_READ, MAP_PRIVATE, fd, 0))<0){
        perror("MMAP error: ");
        exit(EXIT_FAILURE);
    }

    char* buf;
    char char1 = *(memory+1500);

```

```
fprintf(stderr,"Read 1 succeeded past end of file.\n Byte read: %c\tASCII code:%d\n", char1,char1);  
char char2 = *(memory + 5000);  
fprintf(stderr,"Read 2 succeeded past end of file.\n Byte read: %c\tASCII code:%d\n", char2,char2);  
return 0;  
}
```