

ADVANCED UNIX PROGRAMMING REPORT

ASSIGNMENT1



TEAM 9

1. Answer to the questions

Write a program to verify if you can open a file with the append flag to:

(1) Read from the specific place in the file using lseek.

Yes, as you can see, using SEEK_END can read from the specific place in the file successfully.

```

1  int file_descriptor = open(file_path, O_APPEND);
2
3  // Use lseek and read to print "student."
4  off_t read_offset = -8; // Offset to start of "student."
5  lseek(file_descriptor, read_offset, SEEK_END);
6
7  char buffer[8]; // "student." has 8 characters
8  int read_bytes = read(file_descriptor, buffer, sizeof(buffer));

```

```

[mnt/c/Users/Owner/Desktop/Advanced-UNIX-Programming/Hw1 master |2] ✓ system Node 08:13:42 PM 94%
make
gcc -c assignment1.c -o assignment1.o
gcc -std=c11 -O2 -Wall -o assignment1 assignment1.o

[mnt/c/Users/Owner/Desktop/Advanced-UNIX-Programming/Hw1 master |2 ?2] ✓ system Node 08:13:44 PM 94%
./assignment1
student.
Error writing to file: Bad file descriptor

```

(2) Write data at the specific place in the file using lseek.

No, since if the O_APPEND file status flag is set on the open file description, then a write(2) always moves the file offset to the end of the file, regardless of the use of lseek().

```

1  int file_descriptor = open(file_path, O_APPEND);
2  // Use lseek and write to replace "student." with "NTHU student."
3  off_t write_offset = -15; // Offset to start of "student."
4  lseek(file_descriptor, write_offset, SEEK_END);
5
6  const char *replacement = "NTHU student.";
7  int write_bytes = write(file_descriptor, replacement, strlen(replacement));

```

```

[mnt/c/Users/Owner/Desktop/Advanced-UNIX-Programming/Hw1 master |2] ✓ system Node 08:13:42 PM 94%
make
gcc -c assignment1.c -o assignment1.o
gcc -std=c11 -O2 -Wall -o assignment1 assignment1.o

[mnt/c/Users/Owner/Desktop/Advanced-UNIX-Programming/Hw1 master |2 ?2] ✓ system Node 08:13:44 PM 94%
./assignment1
student.
Error writing to file: Bad file descriptor

```

2. Code implementation

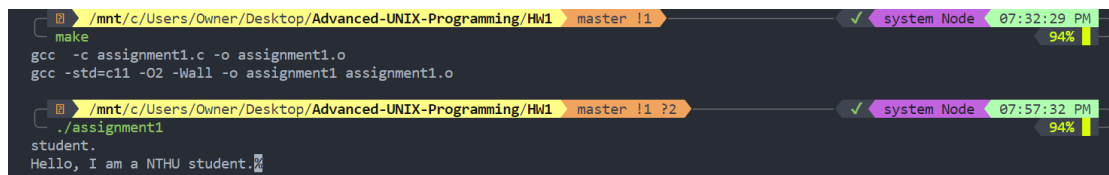
First, we choose O_RDWR to open the file since we need to perform read and write operations.

```
const char *file_path = "sample.txt";

// Open the file in read-write mode
int file_descriptor = open(file_path, O_RDWR);

if (file_descriptor == -1) {
    perror("Failed to open the file");
    return 1;
}
```

The result is output as expected.



The screenshot shows a terminal window with two sessions. The first session shows the compilation of assignment1.c into assignment1.o using gcc, and then the linking of assignment1.o into assignment1 using gcc with flags -std=c11, -O2, and -Wall. The second session shows the execution of ./assignment1, which outputs "student." and "Hello, I am a NTHU student."

```
/mnt/c/Users/Owner/Desktop/Advanced-UNIX-Programming/Hw1 master !1 ✓ system Node 07:32:29 PM 94%
make
gcc -c assignment1.c -o assignment1.o
gcc -std=c11 -O2 -Wall -o assignment1 assignment1.o

/mnt/c/Users/Owner/Desktop/Advanced-UNIX-Programming/Hw1 master !1 ?2 ✓ system Node 07:57:32 PM 94%
./assignment1
student.
Hello, I am a NTHU student.
```