Operating Systems Lab Report

Muhammad Shafeen
Student ID: 22P-9278

October 11, 2024

Lab 8: Operating Systems

0.1 Open 2 Terminals

Opening two terminals checking the process id and then

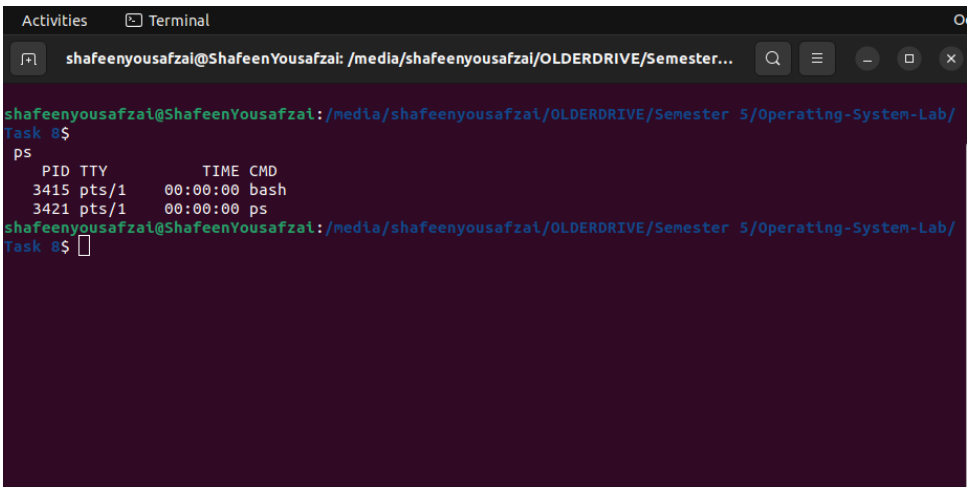

Figure 1: Checking the process id by using PS

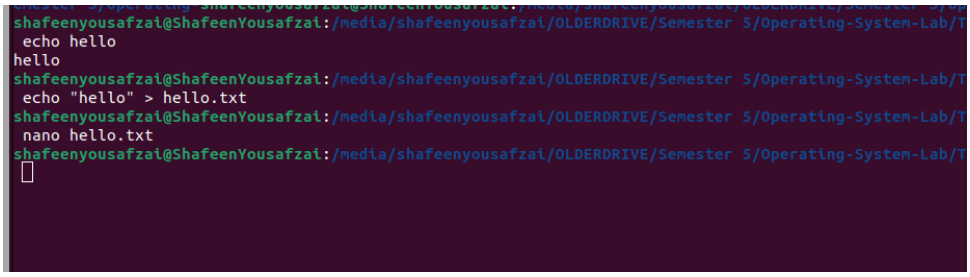

Figure 2: Outputting hello on temrinal using cat ¿

Figure 3: Outputting hello on the proces id termianl

Figure 4: Outputting hello on temrinal using cat ¿

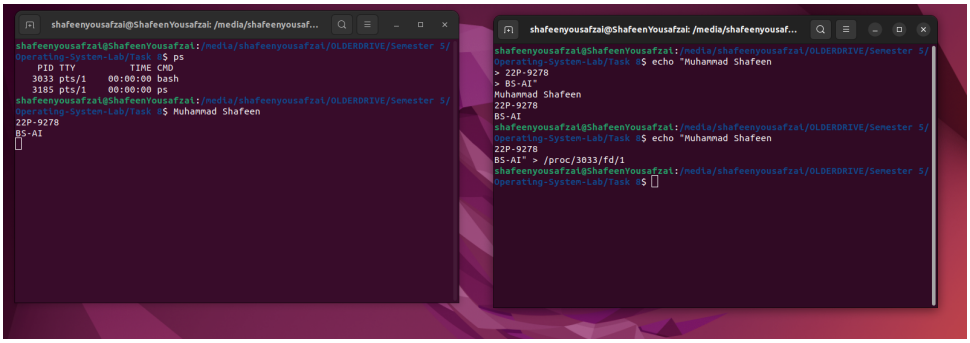

Figure 5: Outputting Name,Roll,section

0.2 Run code and check file size

```
    #include <fcntl.h>
#include <stdio.h>
#include <sys/stat.h>
#include <sys/types.h>
#include <unistd.h>
int main(int argc, char* argv[])
{
char *path = argv[1];
int fd = open(path, O_WRONLY | O_EXCL |
O_CREAT);
if (fd == -1)
{
printf("Error: File not Created\n");
return 1;
```

```
}
return 0;
}
```

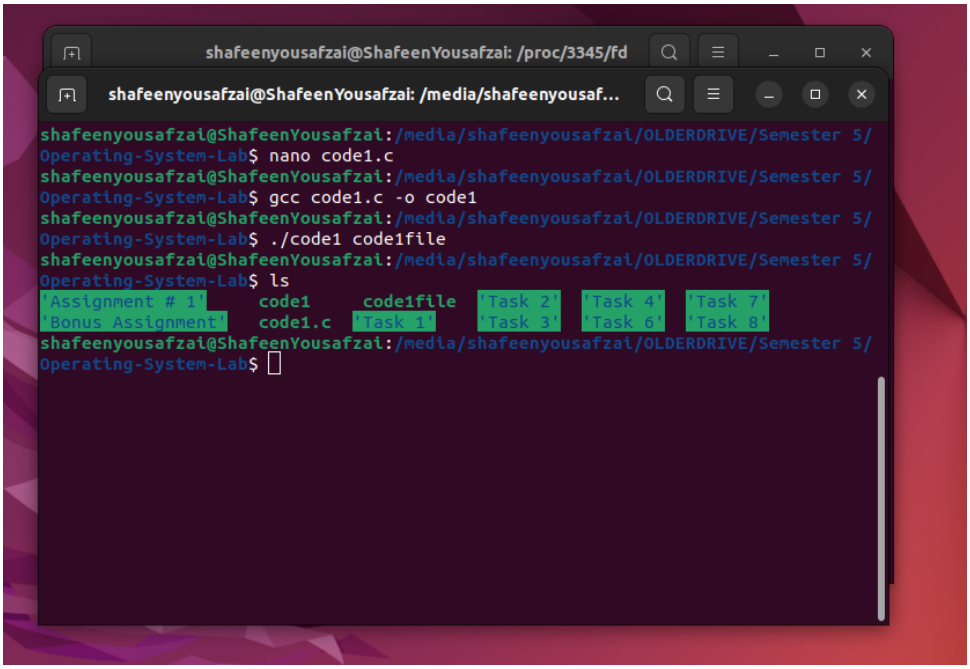

Figure 6: Outputting the output of file and creating a file

0.3 Question

Question What is the size of the file? Why is it this size?

0.4 Answer:

The size of the process is 0 bytes becasue whenever a new file is created it takes 0 bytes of storage

0.5 Question

Check what the open() function return¿

0.6 Answer:

An exit code of 0 indicates success while a non-zero exit code indicates failure. The following failure codes can be returned:

1 Error in command line syntax.
2 One of the files passed on the command line did not exist.
3 A required tool could not be found.
4 The action failed.

0.7 Close a File

```
    #include <fcntl.h>
#include <stdio.h>
#include <sys/stat.h>
#include <sys/types.h>
#include <unistd.h>
int main(int argc, char* argv[])
{
if (argc != 2)
{
printf("Error: Run like this: ");
printf("%6s name-of-new-file\n", argv[0]);
return 1;
}
char *path = argv[1];
int i = 0;
while(i<2)
{
int fd = open(path, O_WRONLY | O_CREAT);
printf("Created! Descriptor is %d\n", fd);
close(fd);
i++;
}
return 0;
}
```

0.8 run without closing file

```
    #include <fcntl.h>
#include <stdio.h>
#include <sys/stat.h>
#include <sys/types.h>
#include <unistd.h>
int main(int argc, char* argv[])
{
if (argc != 2)
{
printf("Error: Run like this: ");
printf("%6s name-of-new-file\n", argv[0]);
return 1;
}
char *path = argv[1];
int i = 0;
while(i<2)
{
int fd = open(path, O_WRONLY | O_CREAT);
printf("Created! Descriptor is %d\n", fd);
i++;
```

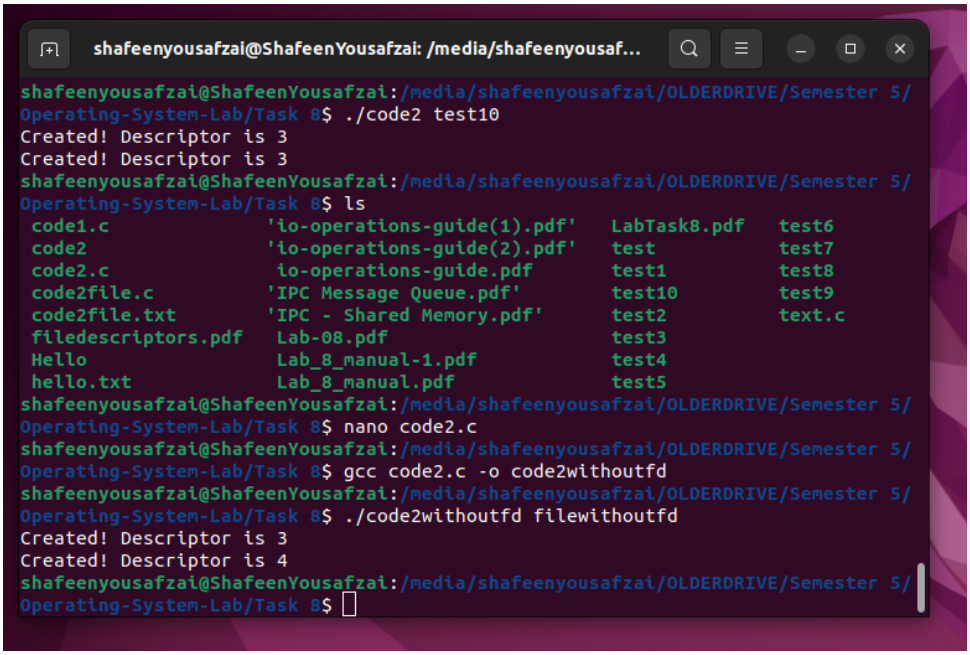

Figure 7: Running the code without fd(close)

```
}
return 0;
}
```

0.9 Question :

Writing to a file is done using the write call. To write, we should obviously open a file first.

0.10 Answer :

we are getting 3 because one file is being run and we get a 4 because we did not close the file created before with a return descriptor of 3 .

0.11 Writing to a file

```
    #include <fcntl.h>
#include <stdio.h>
#include <string.h>
#include <sys/stat.h>
#include <sys/types.h>
#include <time.h>
#include <unistd.h>
char* get_timeStamp()
{
time_t now = time(NULL);
return asctime(localtime(&now));
}
```

```
int main(int argc, char* argv[])
{
char *filename = argv[1];
char *timeStamp = get_timeStamp();
int fd = open(filename, O_WRONLY |
O_APPEND |
O_CREAT, 0666);
size_t length = strlen(timeStamp);
write(fd, timeStamp, length);
close(fd);
return 0;
}
```

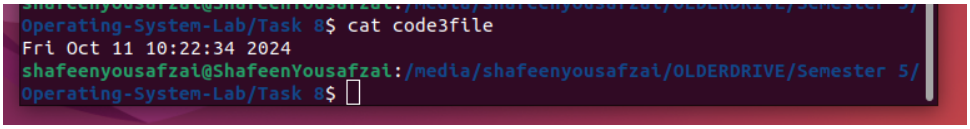

Figure 8: Output of the code. print the output

0.12 Question :

What is 0666 that is specified in the open() call? What does it mean?

0.13 Answer :

0666 is the file permission mode in octal notation. It means the file will be created with read and write permissions for the owner, group, and others.

0.14 Question :

What is O_APPEND doing in the same call? Run the program again and check its output.

0.15 Answer :

O_APPEND flag ensures that the data is appended to the end of the file. If you run the program multiple times, you'll see multiple timestamps in the file, one after another.

0.16 Question :

Modify the following line in the code and then compile and run the pro- gram and check its output. What has happened? From: size_t length = strlen(timeStamp); To: size_t length = strlen(timeStamp)-5

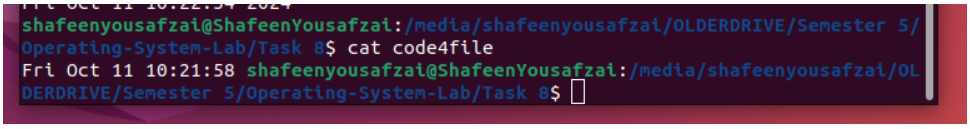

Figure 9: Out of the file with -5 timestamp

0.17 Answer :

By reducing the length by 5, you're truncating the last 5 characters of the times- tamp. This will likely cut off the year and newline character from the timestamp in the file.

```
    #include <stdio.h>
#include <sys/stat.h>
#include <sys/types.h>
#include <unistd.h>
int main(int argc, char* argv[])
{
if (argc != 2)
{
printf("Error: Run like this: ");
printf("%6s name-of-existing-file\n",
argv[0]);
return 1;
}
char *path = argv[1];
int fd = open(path, O_RDONLY);
if (fd == -1)
{
printf("File does not exist\n");
return 1;
}
char buffer[200];
read(fd, buffer, sizeof(buffer)-1);
printf("Contents of File are:\n");
printf("%s\n", buffer);
close(fd);
return 0;
}
```

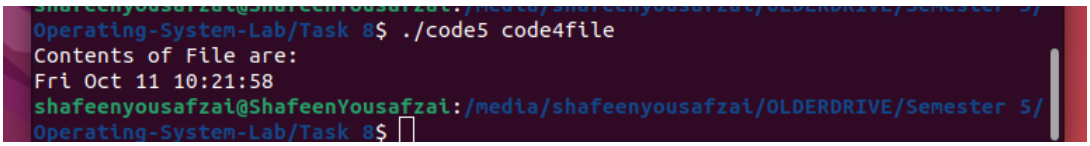

Figure 10: Output of the last code file