Abbas Yadollahi
260680343

ECSE 427

Assignment 1

Part 1

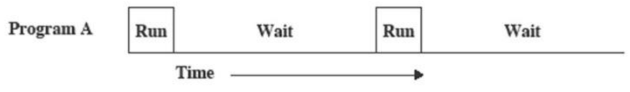
Q1.

	Uniprogramming	Multiprogramming	Timesharing
Memory allocation	There's only 1 program loaded in memory at a given time.	The memory is shared between all programs. It's a lot more memory intensive.	The memory is shared between all programs. It's a lot more memory intensive.
Resource allocation	All resources are given to the single loaded program whether it's running or waiting.	Only 1 program uses the resources at a given time but switches when it doesn't need it anymore.	All programs share the resources by simultaneously using part of the CPU.
Number of programs	Only 1 program runs at a time.	1 program is running. Others are waiting in memory, ready to execute once the one running doesn't require the CPU.	Seems like all programs run at the same time, but the CPU switches in between tasks quickly so it looks simultaneous.

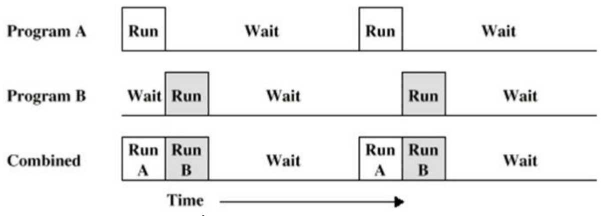
Uniprogramming: In uniprogramming, the computer can only run a single program at a time. Hence, if we had two programs, the first would run from start to finish while the other programs wait until the end, even if the first program is not using the CPU.

Multiprogramming: In multiprogramming, the computer will run one program at a given time, although once it's not using the CPU, another program can start running. Hence, if we had two programs, the first would run until it reaches an IO operation (thus not needing the CPU), where the second program will begin running, thus maximizing the CPU usage.

Timesharing systems: Timesharing is an extension of multiprogramming, where multiple users can have access to the same CPU. The benefit of timesharing systems comes from the lower response times since the CPU constantly switches between all programs, thus elimination idle time and maximizing CPU usage. It works by allocating a short period of computation time to each program and continuously switching between programs.



Uniprogramming system



Multiprogramming system

Q2.

In a uniprogramming system, the programs will run one after the other regardless of the current programs status, thus the time required to run all three programs is 48 ms.

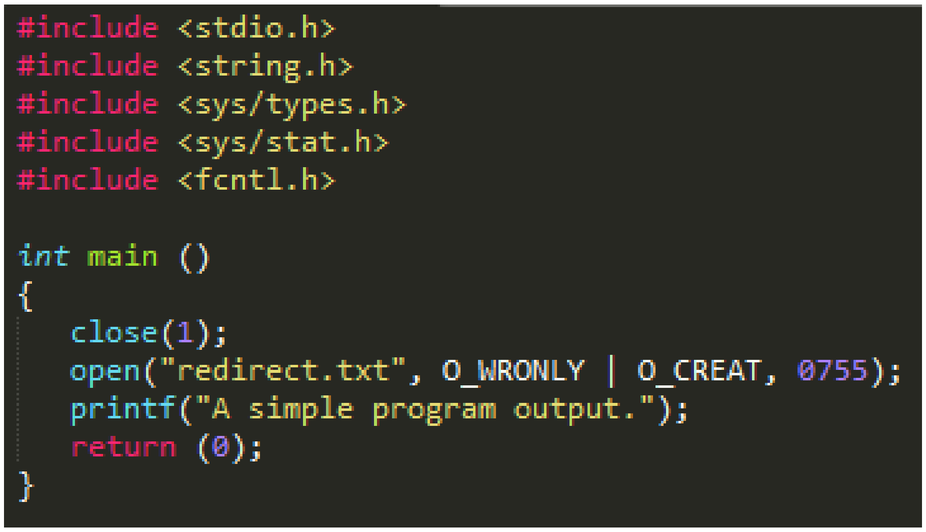
$$(2 + 10 + 4) + (2 + 10 + 4) + (2 + 10 + 4) = 48\,ms$$

In a multiprogramming system, the CPU can run a program while another program is waiting on an IO operation, thus the time required to run all three programs is 24 ms.
Programs B and C run during program A's wait time, meaning there is no time wasted. Program A starts to run again at t=12ms for 4 ms, then program B runs for 4 ms followed by program C for 4 ms.

$$(2 + 10 + 4) + (4) + (4) = 24\,ms$$

Q3.

We need to close the file descriptor 1, which is originally occupied by *stdout* and replace it with a new one directing to the file *redirect.txt*. To do so, we create a new file descriptor by calling *open* with the file name as the first argument, the flags needed as second and the permissions wanted as third.



```c
#include <stdio.h>
#include <string.h>
#include <sys/types.h>
#include <sys/stat.h>
#include <fcntl.h>

int main ()
{
    close(1);
    open("redirect.txt", O_WRONLY | O_CREAT, 0755);
    printf("A simple program output.");
    return (0);
}
```