Network Programming
Exercise set 1

Exercise 4, Child process inheritence

When fork is run, the child process is a exact copy of the parent process, i.e it is the same process but in another memory space. However, some important diferencess remain between the parent process and the child process.

- The child process has its own PID and PPID is the current processes PID
- The child does not inherit the parent processes memory locks, so if the current process has claimed some space in memory, the child process can not access that information
- The child process does not inherit the pending signals that the current process might have
- The semaphore adjustments, meaning that it does not have the same amount of cores/threads to be used as the current process
- The child does inherit open file description locks, meaning that it can use the same files as the current process, however, it does not inherit process-associated record locks.

Exercise 5, Exec system calls

Part a

For all the exec system calls, the program that is being currently run to be replaced by a new process with a newly initialized stack and heap.
For execve(), on top of all the other exec system calls are build on, takes a certain file based on its pathname and the arguments and environment variables are given to it as vectors.
The execl(), execlp() and execle() functions differ in the sense that instead of having the arguments as a vector, they are given straight as pointers to null-terminated strings. So, they are pointers, separated form one another by a comma. This can be visualised by

$$arg0, arg1, \ldots, argn$$

Now, the execl() and execle() differ from each other in the way they get the environment variables. the execle() function gets the environment variables as a vector, just like execve, but execl() gets the environment variables from an external variable *environ*. This is also the case for all other exec functions without the e suffix.
The only difference between execv and execve is that execve gets the environment variables from environ.
All the exec functions that have v after the exec get the arguments as an vector/array.
Now, all the exec functions, that include a p, are not given the executable file as it's pathname, rather they are given the file name and it searches the places listed in the PATH environment variable for the executable file.

Part b

The heap, stack and data segments get changed. Everything else remains the same.

1 Exercise 6, Unix shell commands

Part a

The command

```
ps -aux
```

shows all currently running processes. This can also be done with the flag A, but this gives more information about the running processes.

Part b

To get information about a running process running a certain program, we can utilize the unix command grep

```
ps -aux | grep {program}
```

Part c

1. To kill a certain process, we can utilize the kill command

   ```
   kill {pid}
   ```

2. We can use the pkill command with the f flag

   ```
   pkill -f {program}
   ```

3. Again, using the pkill command, using the u flag, we can kill all of my users processes

   ```
   pkill -u UID
   ```

   UID is a environment variable which holds the UID of the current user. Alternatively, the users username can be used instead of UID.