OPERATING SYSTEMS LAB

ASSIGNMENT – 4

Rahul Cheryala, 210010012

PART – 1

1. The code in minix/servers/sched/schedule.c has been modified, so that "<Roll No> PID <pid> swapped in" is printed whenever a user level process is brought in by the scheduler.

   The code given below has been added in the schedule_process function just above the return statement

Modified Code:

```
if (rmp->priority >= USER_Q)
{
  printf("210010012 PID %d swapped in\n", _ENDPOINT_P(rmp->endpoint));
}
```

2. Run.sh

Written a runme.sh file to copy the schedule.c file to its location and to build the changes.

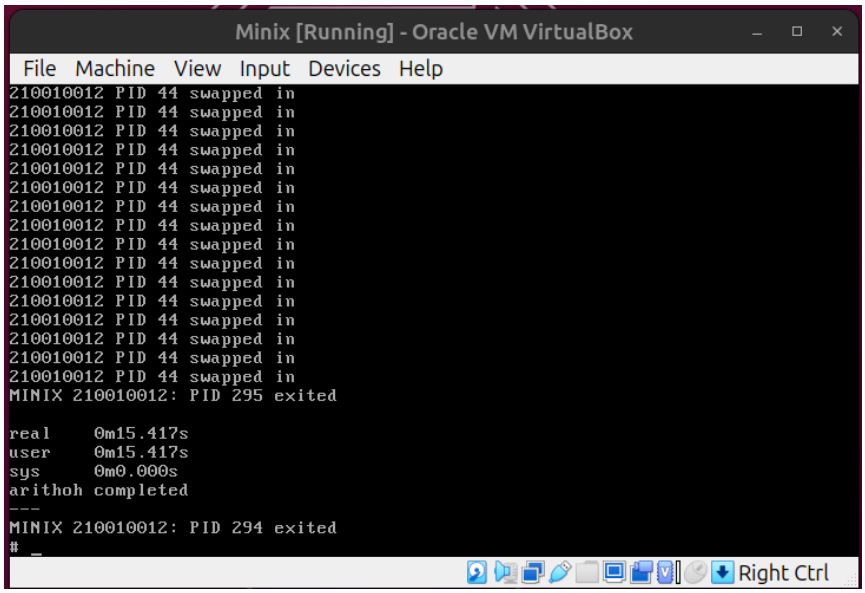
run.sh

210010012 > run.sh

```
cp schedule.c /usr/src/minix/servers/sched/schedule.c
cd /usr/src && make build MKUPDATE=yes
exit 0
```

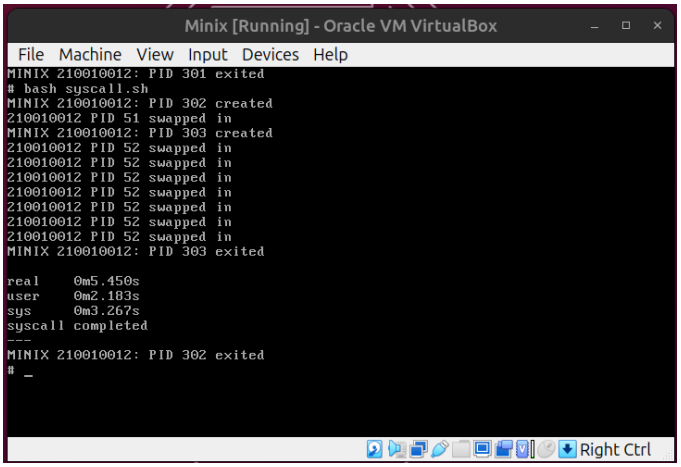
PART – 2

I) arithoh.sh



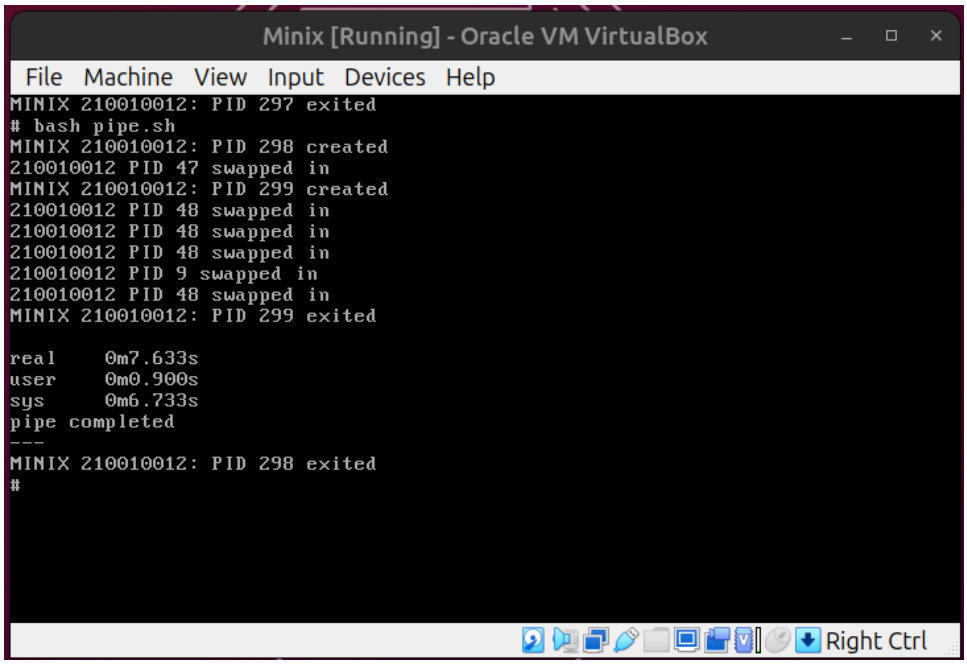
The source code for comprehending the functionality of `arithoh.sh` can be found in the file `UnixBench/src/arith.c`. This script is designed to evaluate the performance of arithmetic operations on a computer system. It achieves this by executing a sequence of arithmetic operations within a loop to assess the system's overall performance. The operations performed in this loop are crucial for gauging the computational capabilities and efficiency of the system.

II) syscall.sh



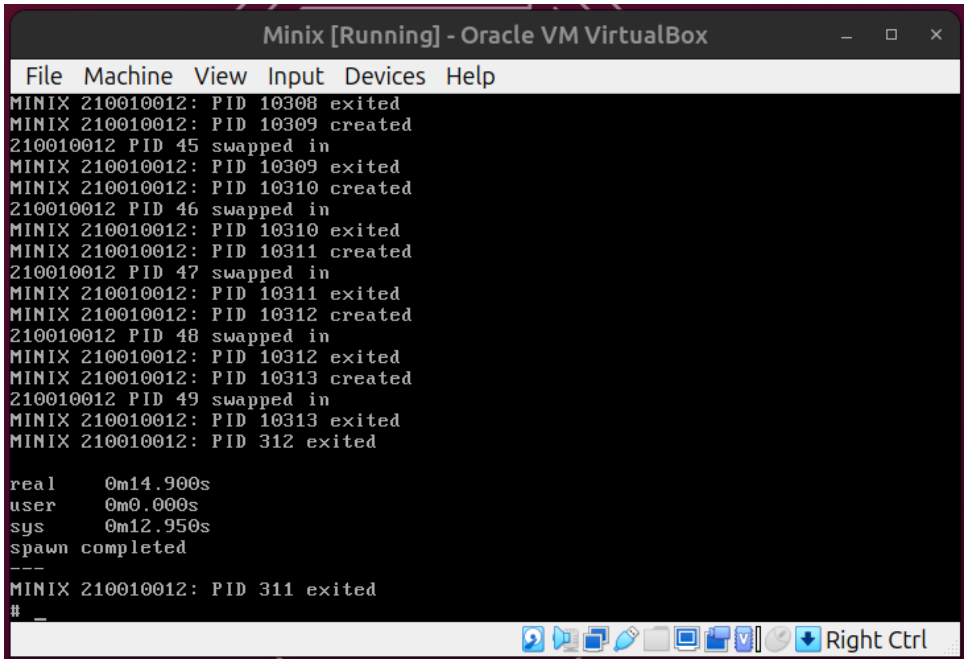
The source code that elucidates the functionality of “syscall.sh” is accessible in the file “UnixBench/src/syscall.c”. This code is designed with the objective of evaluating the performance of system calls by iteratively executing them within a loop. The repetition of these system calls within the loop serves the purpose of measuring and analyzing the efficiency and responsiveness of the system's underlying operating system interfaces.

III) Pipe.sh



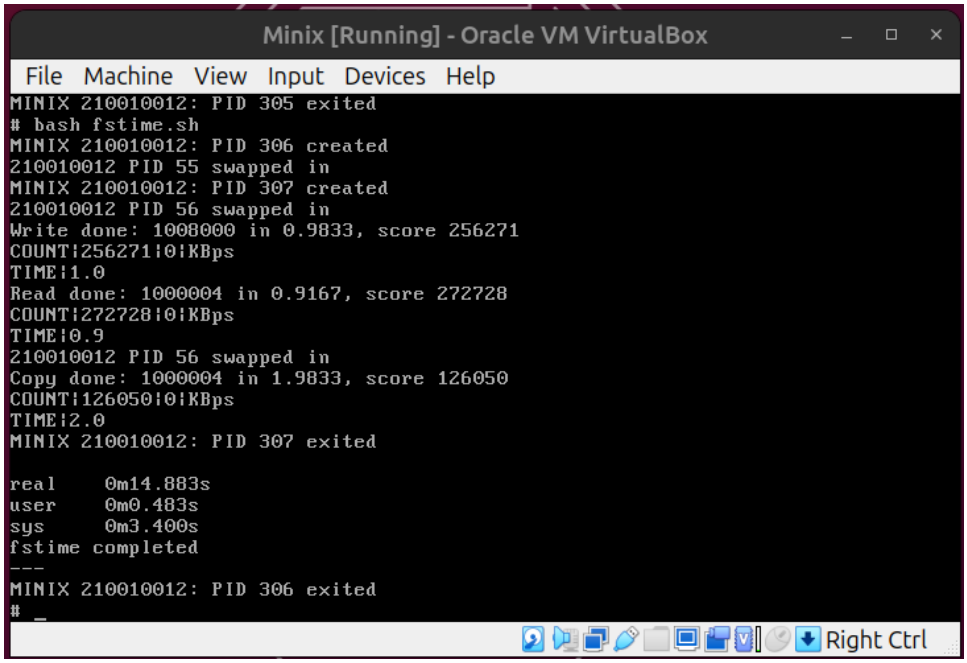
The source code responsible for elucidating the functionality of `pipe.sh` is in the file `UnixBench/src/pipe.c`. This code is crafted with the specific goal of quantifying the throughput of a singular process pipe, all the while avoiding the introduction of context switching. In essence, the code focuses on assessing the efficiency and data transfer capabilities of a single-process pipe without the interference of context switches.

IV) Spawn.sh



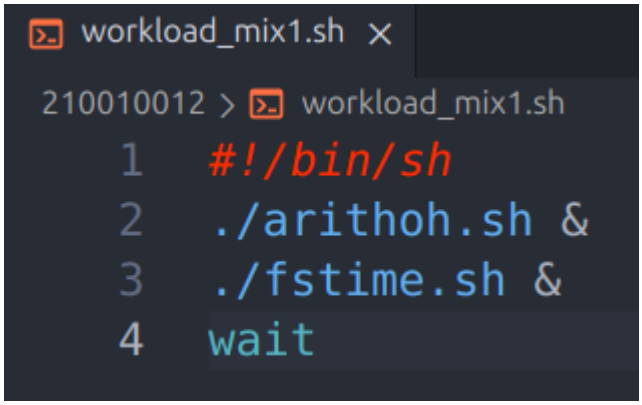
The source code that provides insight into the functioning of `spawn.sh` can be found in the file `UnixBench/src/spawn.c`. This code is designed to evaluate the performance of creating and immediately terminating child processes within a loop. The objective is to assess the efficiency and speed of the process creation and termination operations in a repetitive manner, allowing for a comprehensive measurement of the system's capability in handling such tasks.

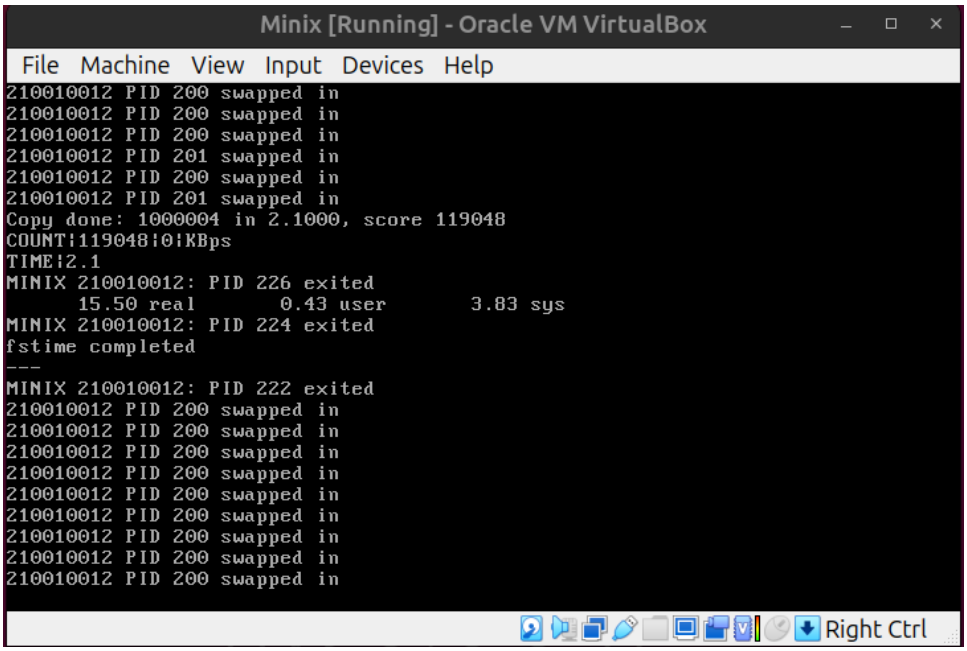
V) Fstime.sh

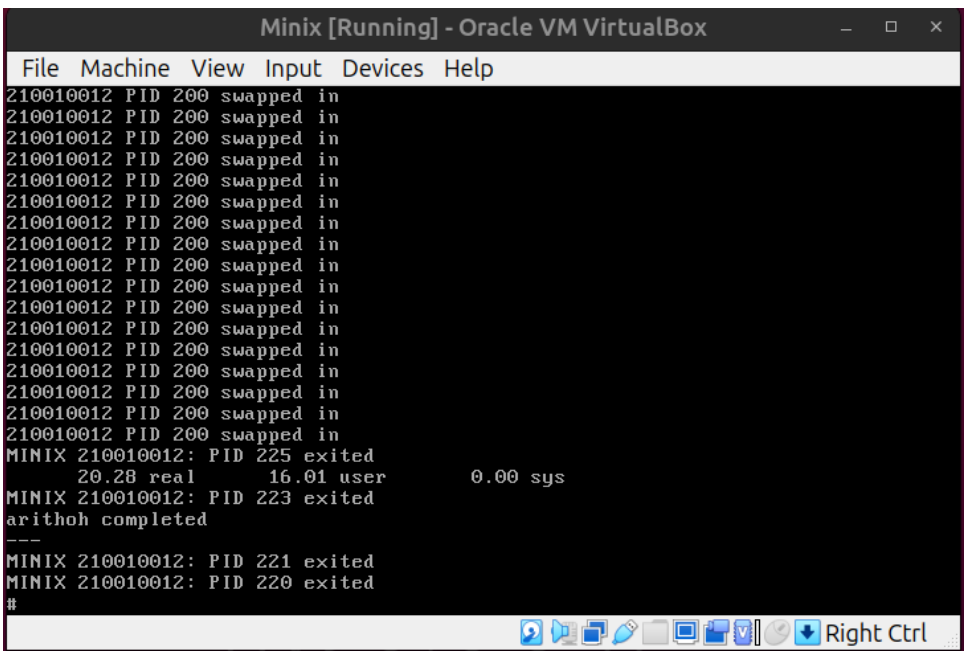


The source code providing an understanding of the workings of `fstime.sh` is located in the file `UnixBench/src/fstime.c`. The primary purpose of this code is to gauge file system performance by conducting tests associated with reading, writing, and copying data. In essence, it assesses the efficiency of the file system operations, offering insights into the system's capabilities when it comes to handling data-related tasks.

Workload_mix1.sh :







In summary, based on the code analysis and PID printing order:

- `arithoh` is a CPU-bound process because it involves arithmetic operations, which are typically CPU-intensive.

- `fstime` is an I/O-bound process as it focuses on file operations, such as reading, writing, and copying data, which are often associated with waiting

for I/O operations to complete.

Workload_mix2.sh :

workload_mix2.sh

210010012 > workload_mix2.sh

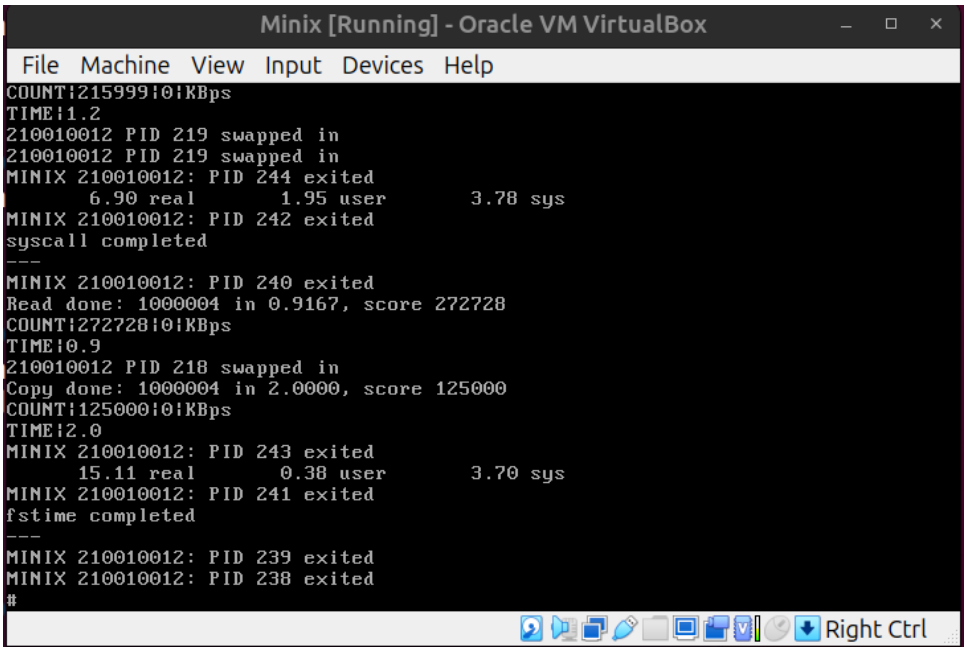
```
#!/bin/sh
./fstime.sh &
./syscall.sh &
wait
```

Minix [Running] - Oracle VM VirtualBox

File Machine View Input Devices Help

```
MINIX 210010012: PID 237 exited
# bash workload_mix2.sh
MINIX 210010012: PID 238 created
210010012 PID 213 swapped in
MINIX 210010012: PID 239 created
210010012 PID 214 swapped in
MINIX 210010012: PID 240 created
210010012 PID 215 swapped in
MINIX 210010012: PID 241 created
210010012 PID 216 swapped in
MINIX 210010012: PID 242 created
210010012 PID 217 swapped in
MINIX 210010012: PID 243 created
210010012 PID 218 swapped in
MINIX 210010012: PID 244 created
210010012 PID 219 swapped in
210010012 PID 219 swapped in
210010012 PID 219 swapped in
210010012 PID 219 swapped in
210010012 PID 219 swapped in
Write done: 1008000 in 1.1667, score 215999
COUNT|215999|0|KBps
TIME|1.2
210010012 PID 219 swapped in
```

Right Ctrl



Based on the analysis of the code and the order of Process IDs (PIDs) printing:

- `fstime` is confirmed as an I/O Bound process, as it involves file operations.

- For `syscall`, the presence of mix tests, getpid tests, and exec tests suggests a mix of CPU-bound and I/O-bound characteristics. The mix tests, getpid tests, and exec tests contribute to its CPU-bound nature, while the close tests make it exhibit I/O-bound characteristics. In summary, `syscall` is a hybrid or mixed-process, combining elements of both CPU-bound and I/O-bound behaviors.

Workload_mix3.sh :

workload_mix3.sh

210010012 > workload_mix3.sh

```
#!/bin/sh
./arithoh.sh &
./spawn.sh &
wait
```

Minix [Running] - Oracle VM VirtualBox

File Machine View Input Devices Help

```
MINIX 210010012: PID 251 exited
      15.90 real       0.00 user       9.65 sys
MINIX 210010012: PID 249 exited
spawn completed
---
MINIX 210010012: PID 247 exited
210010012 PID 225 swapped in
210010012 PID 225 swapped in
210010012 PID 225 swapped in
210010012 PID 225 swapped in
210010012 PID 225 swapped in
210010012 PID 225 swapped in
210010012 PID 225 swapped in
210010012 PID 225 swapped in
210010012 PID 225 swapped in
210010012 PID 225 swapped in
210010012 PID 225 swapped in
210010012 PID 225 swapped in
210010012 PID 225 swapped in
210010012 PID 225 swapped in
210010012 PID 225 swapped in
210010012 PID 225 swapped in
210010012 PID 225 swapped in
210010012 PID 225 swapped in
```

Right Ctrl

Minix [Running] - Oracle VM VirtualBox

File Machine View Input Devices Help

```
210010012 PID 225 swapped in
210010012 PID 225 swapped in
210010012 PID 225 swapped in
210010012 PID 225 swapped in
210010012 PID 225 swapped in
210010012 PID 225 swapped in
210010012 PID 225 swapped in
210010012 PID 225 swapped in
210010012 PID 225 swapped in
210010012 PID 225 swapped in
210010012 PID 225 swapped in
210010012 PID 225 swapped in
210010012 PID 225 swapped in
210010012 PID 225 swapped in
210010012 PID 225 swapped in
210010012 PID 225 swapped in
210010012 PID 225 swapped in
MINIX 210010012: PID 250 exited
      30.43 real      15.40 user       0.00 sys
MINIX 210010012: PID 248 exited
arithoh completed
---
MINIX 210010012: PID 246 exited
MINIX 210010012: PID 245 exited
# _
```

Right Ctrl

Based on the analysis of the code and the order in which Process IDs (PIDs) are printed:

- `arithoh` is confirmed as a CPU Bound Process due to its involvement in arithmetic operations, which are typically CPU-intensive.

- For `spawn.sh`, the fact that it deals with process creation and termination in a loop supports the assertion that it is also a CPU Bound Process. The repetitive creation and termination of processes involve significant CPU utilization, aligning with the characteristics of CPU-bound tasks.

In summary, `spawn.sh` is considered a CPU Bound Process based on its code analysis and the nature of process creation and termination operations in the loop.