

# 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_0)
{
    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
1  cp schedule.c /usr/src/minix/servers/sched/schedule.c
2  cd /usr/src && make build MKUPDATE=yes
3  exit 0
```

## PART – 2

l) arithoh.sh

The screenshot shows the Oracle VM VirtualBox interface. The title bar reads "Minix [Running] - Oracle VM VirtualBox". Below it is a menu bar with "File", "Machine", "View", "Input", "Devices", and "Help". The main area contains a terminal window with the following output:  

```
Z10010012 PID 44 swapped in  
Z10010012 PID 44 swapped in  
Z10010012 PID 44 swapped in  
Z10010012 PID 44 swapped in  
Z10010012 PID 44 swapped in  
Z10010012 PID 44 swapped in  
Z10010012 PID 44 swapped in  
Z10010012 PID 44 swapped in  
Z10010012 PID 44 swapped in  
Z10010012 PID 44 swapped in  
Z10010012 PID 44 swapped in  
Z10010012 PID 44 swapped in  
Z10010012 PID 44 swapped in  
Z10010012 PID 44 swapped in  
Z10010012 PID 44 swapped in  
MINIX Z10010012: PID 295 exited
```

  
Below this, there are statistics:  

```
real    0m15.417s  
user    0m15.417s  
sys     0m0.000s  
arithoh completed
```

  
Then another exit message:  

```
---  
MINIX Z10010012: PID 294 exited
```

  
At the bottom, a prompt character "#" is visible. The Windows taskbar at the very bottom shows several icons and the text "Right Ctrl".

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

```
Minix [Running] - Oracle VM VirtualBox
```

```
File Machine View Input Devices Help
```

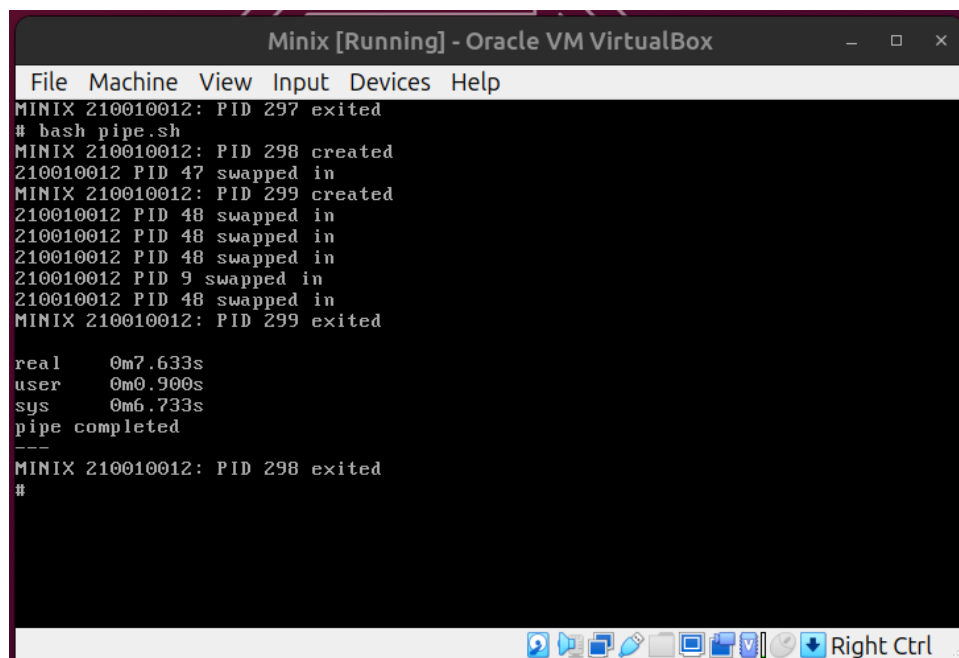
```
MINIX 210010012: PID 301 exited
# bash syscall.sh
MINIX 210010012: PID 302 created
210010012 PID 51 swapped in
MINIX 210010012: PID 303 created
210010012 PID 52 swapped in
210010012 PID 52 swapped in
210010012 PID 52 swapped in
210010012 PID 52 swapped in
210010012 PID 52 swapped in
210010012 PID 52 swapped in
210010012 PID 52 swapped in
MINIX 210010012: PID 303 exited

real    0m5.450s
user    0m2.183s
sys      0m3.267s
syscall completed

MINIX 210010012: PID 302 exited
# _
```

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

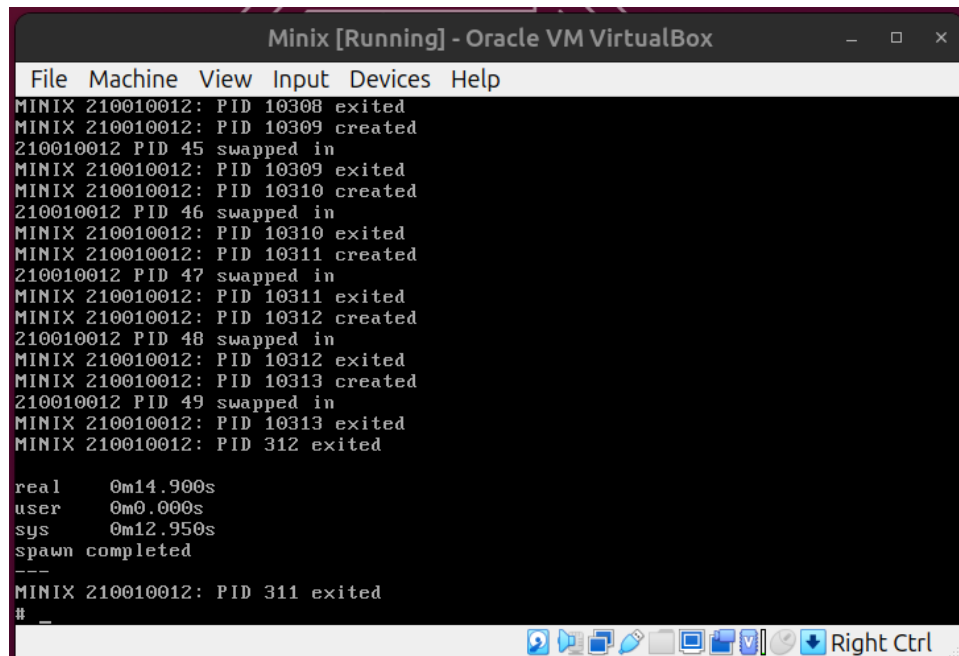
A screenshot of a Minix terminal window titled "Minix [Running] - Oracle VM VirtualBox". The window has a menu bar with "File", "Machine", "View", "Input", "Devices", and "Help". The terminal output shows the execution of the "pipe.sh" script. It starts with "MINIX 210010012: PID 297 exited", followed by "# bash pipe.sh". Then, "MINIX 210010012: PID 298 created" is shown, followed by a series of "210010012 PID 47 swapped in", "MINIX 210010012: PID 299 created", and several "210010012 PID 48 swapped in" and "210010012 PID 9 swapped in" messages. The script then reports timing: "real 0m7.633s", "user 0m0.900s", and "sys 0m6.733s", followed by "pipe completed". After a separator "---", it shows "MINIX 210010012: PID 298 exited" and a prompt "#". The bottom of the window features a toolbar with various icons and a "Right Ctrl" button.

```
Minix [Running] - Oracle VM VirtualBox
File Machine View Input Devices Help
MINIX 210010012: PID 297 exited
# bash pipe.sh
MINIX 210010012: PID 298 created
210010012 PID 47 swapped in
MINIX 210010012: PID 299 created
210010012 PID 48 swapped in
210010012 PID 48 swapped in
210010012 PID 48 swapped in
210010012 PID 9 swapped in
210010012 PID 48 swapped in
MINIX 210010012: PID 299 exited

real    0m7.633s
user    0m0.900s
sys      0m6.733s
pipe completed
---
MINIX 210010012: PID 298 exited
#
```

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 screenshot shows a terminal window titled "Minix [Running] - Oracle VM VirtualBox". The terminal displays the output of a script that creates and terminates child processes in a loop. The output includes the following lines:

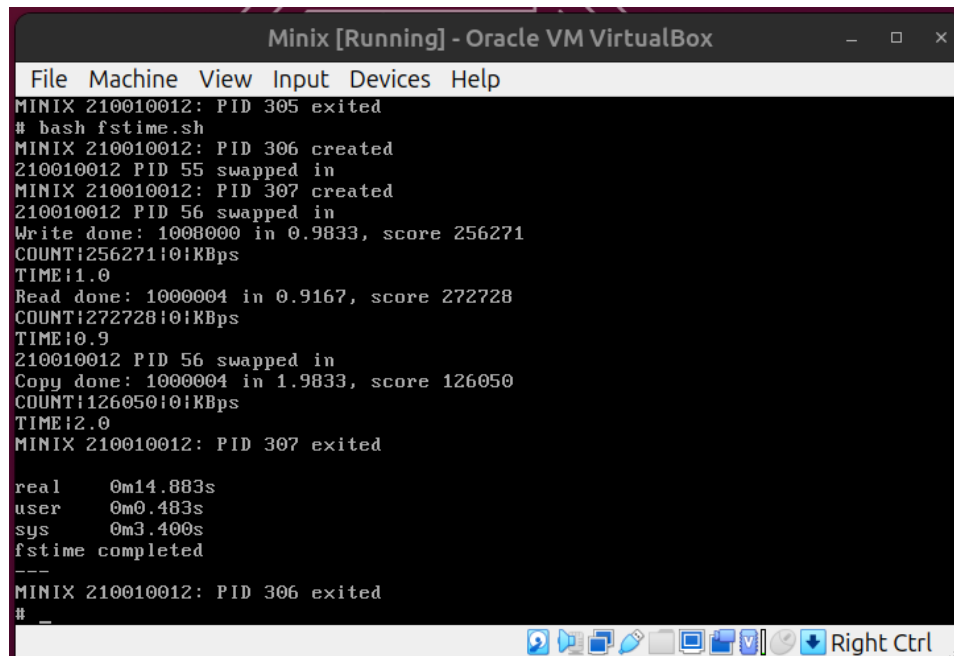
```
File Machine View Input Devices Help
MINIX 210010012: PID 10308 exited
MINIX 210010012: PID 10309 created
210010012 PID 45 swapped in
MINIX 210010012: PID 10309 exited
MINIX 210010012: PID 10310 created
210010012 PID 46 swapped in
MINIX 210010012: PID 10310 exited
MINIX 210010012: PID 10311 created
210010012 PID 47 swapped in
MINIX 210010012: PID 10311 exited
MINIX 210010012: PID 10312 created
210010012 PID 48 swapped in
MINIX 210010012: PID 10312 exited
MINIX 210010012: PID 10313 created
210010012 PID 49 swapped in
MINIX 210010012: PID 10313 exited
MINIX 210010012: PID 312 exited

real    0m14.900s
user    0m0.000s
sys     0m12.950s
spawn completed
---
MINIX 210010012: PID 311 exited
# _
```

The terminal window also shows a status bar at the bottom with icons for various functions and a "Right Ctrl" button.

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

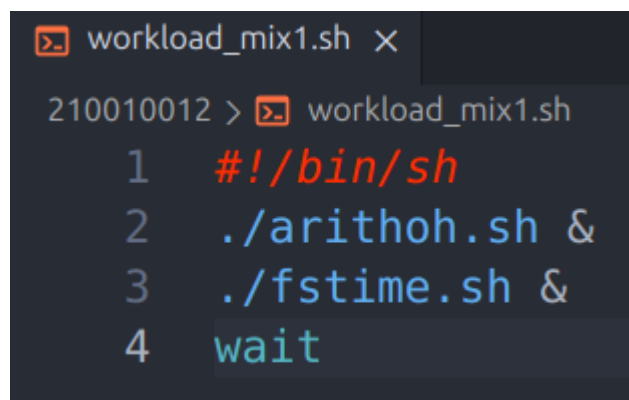


```
Minix [Running] - Oracle VM VirtualBox
File Machine View Input Devices Help
MINIX 210010012: PID 305 exited
# bash fstime.sh
MINIX 210010012: PID 306 created
210010012 PID 55 swapped in
MINIX 210010012: PID 307 created
210010012 PID 56 swapped in
Write done: 1000000 in 0.9833, score 256271
COUNT:256271:0:KBps
TIME:1.0
Read done: 1000000 in 0.9167, score 272728
COUNT:272728:0:KBps
TIME:0.9
210010012 PID 56 swapped in
Copy done: 1000000 in 1.9833, score 126050
COUNT:126050:0:KBps
TIME:2.0
MINIX 210010012: PID 307 exited

real    0m14.883s
user    0m0.483s
sys     0m3.400s
fstime completed
---
MINIX 210010012: PID 306 exited
# _
```

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 :



```
workload_mix1.sh x
210010012 > workload_mix1.sh
1  #!/bin/sh
2  ./arithoh.sh &
3  ./fstime.sh &
4  wait
```

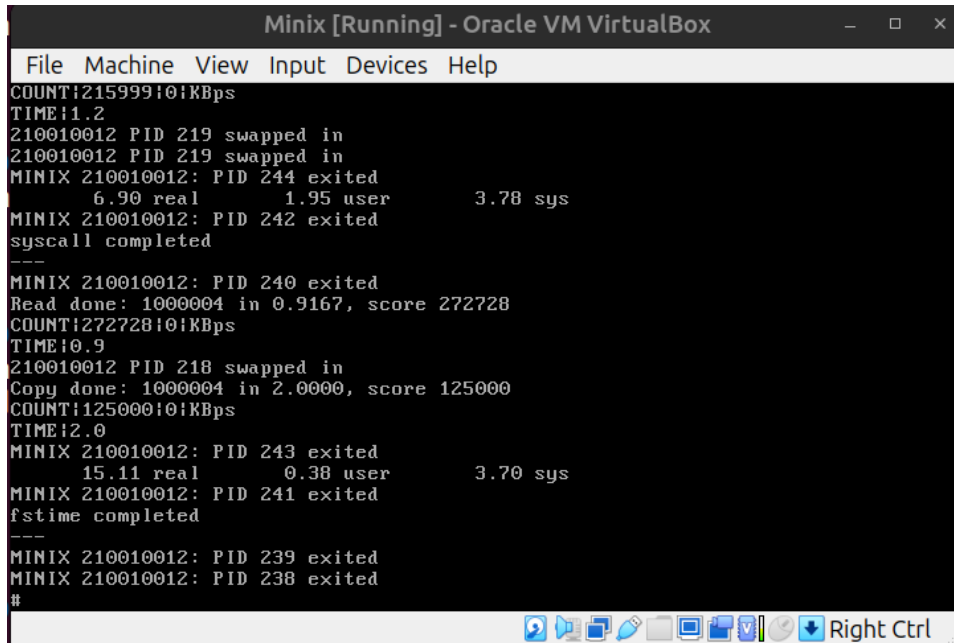


for I/O operations to complete.

### Workload\_mix2.sh :

```
workload_mix2.sh ●
210010012 > workload_mix2.sh
1  #!/bin/sh
2  ./fstime.sh &
3  ./syscall.sh &
4  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:215999:0KBps
TIME:1.2
210010012 PID 219 swapped in
Right Ctrl
```



```
Minix [Running] - Oracle VM VirtualBox
File Machine View Input Devices Help
COUNT:215999:0:KBps
TIME:1.2
210010012 PID 219 swapped in
210010012 PID 219 swapped in
MINIX 210010012: PID 244 exited
        6.90 real        1.95 user        3.78 sys
MINIX 210010012: PID 242 exited
syscall completed
---
MINIX 210010012: PID 240 exited
Read done: 1000004 in 0.9167, score 272728
COUNT:272728:0:KBps
TIME:0.9
210010012 PID 218 swapped in
Copy done: 1000004 in 2.0000, score 125000
COUNT:125000:0:KBps
TIME:2.0
MINIX 210010012: PID 243 exited
        15.11 real        0.38 user        3.70 sys
MINIX 210010012: PID 241 exited
fstime completed
---
MINIX 210010012: PID 239 exited
MINIX 210010012: PID 238 exited
#
```

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 x
210010012 > workload_mix3.sh
1  #!/bin/sh
2  ./arithoh.sh &
3  ./spawn.sh &
4  wait
```

The screenshot shows the Oracle VM VirtualBox interface. The title bar reads "Minix [Running] - Oracle VM VirtualBox". Below the title bar is a menu bar with "File", "Machine", "View", "Input", "Devices", and "Help". The main area contains a terminal window displaying the following text:  

```
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
```

  
At the bottom of the screen is a taskbar with various icons and the text "Right Ctrl".A screenshot of a virtual machine window titled "Minix [Running] - Oracle VM VirtualBox". The window has a menu bar with "File", "Machine", "View", "Input", "Devices", and "Help". The main area displays the output of a Minix boot process. It shows multiple instances of "PID 225 swapped in" followed by "MINIX PID 250 exited", "30.43 real 15.40 user 0.00 sys", "MINIX PID 248 exited", "arithoh completed", three dashes "---", "MINIX PID 246 exited", "MINIX PID 245 exited", and finally a prompt "#". At the bottom of the window, there is a taskbar with several icons and the text "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		
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.