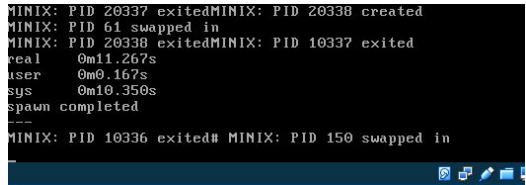
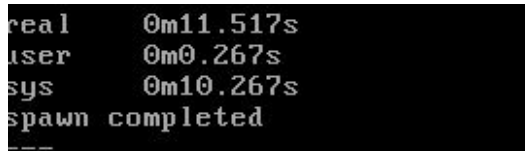
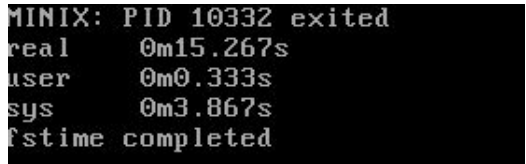


# OS LAB 4

Janhavi Dadhania

Roll No.: 180010012

1	Preparing workload Mixes: Using workload mixes from UnixBench Suite, modified to spawn 5 processes						
	<p>The unixBench suite workload_mixes files are modified to spawn 5 processes each covering i/o intense , computationally intensive processes. Modified files added in the folder.</p> <p>4 files are :</p> <ol style="list-style-type: none"><li>1.spawn.c</li><li>2.fstime.c</li><li>3.syscall.c</li><li>4.arith.c</li></ol> <p>Their execution with a priority based round-robin scheduler is as below.</p> <table><tr><td>syscall.c</td><td><pre>real    0m5.700s user    0m1.783s sys     0m3.900s syscall completed ---</pre><p>Time spent in each mode for syscall.c is as above.</p><pre>MINIX: PID 10331 exited# bash syscall.sh MINIX: PID 10333 created MINIX: PID 69 swapped in MINIX: PID 10334 created MINIX: PID 70 swapped in MINIX: PID 70 swapped in MINIX: PID 70 swapped in MINIX: PID 70 swapped in MINIX: PID 70 swapped in MINIX: PID 70 swapped in MINIX: PID 70 swapped in MINIX: PID 10334 exited real    0m5.700s user    0m1.783s sys     0m3.900s syscall completed ---</pre><p>As seen above, process 10332 is not exited and 10333 and 10334 and 10331 are exited. This is because the scheduler implements priority based round robin scheduling policy.</p></td></tr><tr><td></td><td><table><tr><td>arith.c</td><td><pre>real    0m17.733s user    0m17.700s sys     0m0.017s arithoh completed</pre><p>time spent in kernel mode is very less compared to user mode for arith.c</p></td></tr></table></td></tr></table>	syscall.c	<pre>real    0m5.700s user    0m1.783s sys     0m3.900s syscall completed ---</pre> <p>Time spent in each mode for syscall.c is as above.</p> <pre>MINIX: PID 10331 exited# bash syscall.sh MINIX: PID 10333 created MINIX: PID 69 swapped in MINIX: PID 10334 created MINIX: PID 70 swapped in MINIX: PID 70 swapped in MINIX: PID 70 swapped in MINIX: PID 70 swapped in MINIX: PID 70 swapped in MINIX: PID 70 swapped in MINIX: PID 70 swapped in MINIX: PID 10334 exited real    0m5.700s user    0m1.783s sys     0m3.900s syscall completed ---</pre> <p>As seen above, process 10332 is not exited and 10333 and 10334 and 10331 are exited. This is because the scheduler implements priority based round robin scheduling policy.</p>		<table><tr><td>arith.c</td><td><pre>real    0m17.733s user    0m17.700s sys     0m0.017s arithoh completed</pre><p>time spent in kernel mode is very less compared to user mode for arith.c</p></td></tr></table>	arith.c	<pre>real    0m17.733s user    0m17.700s sys     0m0.017s arithoh completed</pre> <p>time spent in kernel mode is very less compared to user mode for arith.c</p>
syscall.c	<pre>real    0m5.700s user    0m1.783s sys     0m3.900s syscall completed ---</pre> <p>Time spent in each mode for syscall.c is as above.</p> <pre>MINIX: PID 10331 exited# bash syscall.sh MINIX: PID 10333 created MINIX: PID 69 swapped in MINIX: PID 10334 created MINIX: PID 70 swapped in MINIX: PID 70 swapped in MINIX: PID 70 swapped in MINIX: PID 70 swapped in MINIX: PID 70 swapped in MINIX: PID 70 swapped in MINIX: PID 70 swapped in MINIX: PID 10334 exited real    0m5.700s user    0m1.783s sys     0m3.900s syscall completed ---</pre> <p>As seen above, process 10332 is not exited and 10333 and 10334 and 10331 are exited. This is because the scheduler implements priority based round robin scheduling policy.</p>						
	<table><tr><td>arith.c</td><td><pre>real    0m17.733s user    0m17.700s sys     0m0.017s arithoh completed</pre><p>time spent in kernel mode is very less compared to user mode for arith.c</p></td></tr></table>	arith.c	<pre>real    0m17.733s user    0m17.700s sys     0m0.017s arithoh completed</pre> <p>time spent in kernel mode is very less compared to user mode for arith.c</p>				
arith.c	<pre>real    0m17.733s user    0m17.700s sys     0m0.017s arithoh completed</pre> <p>time spent in kernel mode is very less compared to user mode for arith.c</p>						

		<p>executable file.</p>  <p>As it can be seen above process 10336 is exited as last i.e processes 10337 exits before it.</p>
	spawn.c	 <p>In this case(spawn.c) time spent in sys mode is greater than in user mode opposite to arith.c file.</p>
	fstime.c	 <p>For fstime.c time spent in user and sys mode is as above.</p>
2	<b>Pseudo FIFO</b>	
	<p>“among the user-level processes that are ready to execute, the one that entered the earliest must be scheduled.”</p> <p><b>Implementation</b></p> <p>Priority value (stored in variable rmp-&gt;priority) is increased and decreased in the functions do_noquantum and balance_queues in schedule.c file. At the start, all processes priority values are initialised in enqueue function in the proc.c file. Thus, the increase and decrease in priority values ensures that high priority processes are scheduled first.</p> <p>In FIFO we want to remove the priority ordering and schedule the process which came first and is ready. Thus, increase and decrease in the priority value needs to be deleted.</p> <p>Thus, the lines which increase and decrease the rp-&gt;p_priority values are commented in the code as below.</p>	

```
static void balance_queues(minix_timer_t *tp)
{
    struct schedproc *rmp;
    int proc_nr;

    for (proc_nr=0, rmp=schedproc; proc_nr < NR_PROCS; proc_nr++, rmp++) {
        if (rmp->flags & IN_USE) {
            if (rmp->priority > rmp->max_priority) {
                rmp->priority -= 1; /* increase priority */
                schedule_process_local(rmp);
            }
        }
    }
}
```

```
int do_noquantum(message *m_ptr)
{
    register struct schedproc *rmp;
    int rv, proc_nr_n;

    if (sched_isokendpt(m_ptr->m_source, &proc_nr_n) != OK) {
        printf("SCHED: WARNING: got an invalid endpoint in 00Q msg %u.\n",
            m_ptr->m_source);
        return EBADEPT;
    }

    rmp = &schedproc[proc_nr_n];
    if (rmp->priority < MIN_USER_Q) {
        rmp->priority += 1; /* lower priority */
    }

    if ((rv = schedule_process_local(rmp)) != OK) {
        return rv;
    }
    return OK;
}
```

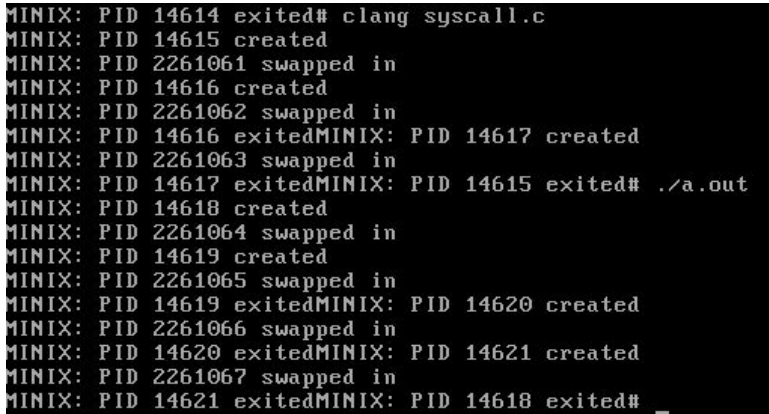
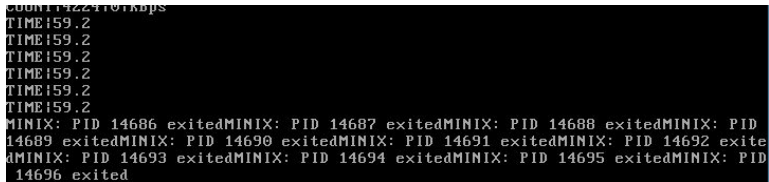
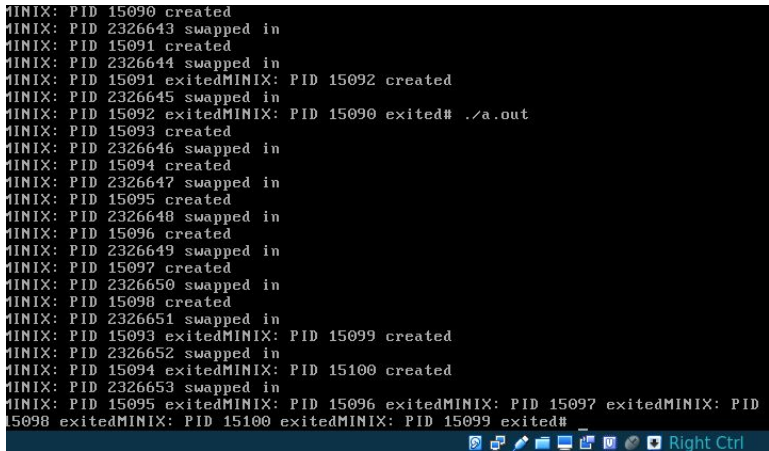
## Workload behaviour under new FIFO

Modified c files are present in the folder

spawn.c

```
MINIX: PID 10233 created
MINIX: PID 1573059 swapped in
MINIX: PID 10234 created
MINIX: PID 1573060 swapped in
MINIX: PID 10234 exitedMINIX: PID 10235 created
MINIX: PID 1573061 swapped in
MINIX: PID 10235 exitedMINIX: PID 10236 created
MINIX: PID 1573062 swapped in
MINIX: PID 10236 exitedMINIX: PID 10237 created
MINIX: PID 1573063 swapped in
MINIX: PID 10237 exitedMINIX: PID 10238 created
MINIX: PID 1573064 swapped in
MINIX: PID 10238 exitedMINIX: PID 10239 created
MINIX: PID 1573065 swapped in
MINIX: PID 10239 exitedMINIX: PID 10233 exited#
```

As seen in the image above, process 10235 is created and it exits first.  
Then, process 10236 is created and it exits.  
After that process 10237 is created which exits before process 10238 is created.  
Process 10238 in turn exits before 10239 is created.

	<p>Thus, the process which came first is scheduled first and is thus exited first.</p>
syscall.c	 <pre> MINIX: PID 14614 exited# clang syscall.c MINIX: PID 14615 created MINIX: PID 2261061 swapped in MINIX: PID 14616 created MINIX: PID 2261062 swapped in MINIX: PID 14616 exitedMINIX: PID 14617 created MINIX: PID 2261063 swapped in MINIX: PID 14617 exitedMINIX: PID 14615 exited# ./a.out MINIX: PID 14618 created MINIX: PID 2261064 swapped in MINIX: PID 14619 created MINIX: PID 2261065 swapped in MINIX: PID 14619 exitedMINIX: PID 14620 created MINIX: PID 2261066 swapped in MINIX: PID 14620 exitedMINIX: PID 14621 created MINIX: PID 2261067 swapped in MINIX: PID 14621 exitedMINIX: PID 14618 exited# </pre> <p>syscall.c is modified to spawn 5 different processes which could be seen above.</p> <p>Here, in case of FIFO, after a.out is executed, process 14619 is created which first child process and it exits first. Then, the next child process 14620 is created and is exited first because it was scheduled first under FIFO policy.</p>
fstime.c	 <pre> 000114224101kaps TIME:59.2 TIME:59.2 TIME:59.2 TIME:59.2 TIME:59.2 TIME:59.2 MINIX: PID 14686 exitedMINIX: PID 14687 exitedMINIX: PID 14688 exitedMINIX: PID 14689 exitedMINIX: PID 14690 exitedMINIX: PID 14691 exitedMINIX: PID 14692 exite dMINIX: PID 14693 exitedMINIX: PID 14694 exitedMINIX: PID 14695 exitedMINIX: PID 14696 exited </pre> <p>Screenshot of execution of syscall.c executable end.</p> <p>As it could be seen in the image above the processes ends in the order 14686, 14687, 14688, 14689, 14690, 14691, 14692, 14693, 14694, 14695, 14696 which is as expected according to FIFO policy.</p>
arith.c	 <pre> MINIX: PID 15090 created MINIX: PID 2326643 swapped in MINIX: PID 15091 created MINIX: PID 2326644 swapped in MINIX: PID 15091 exitedMINIX: PID 15092 created MINIX: PID 2326645 swapped in MINIX: PID 15092 exitedMINIX: PID 15090 exited# ./a.out MINIX: PID 15093 created MINIX: PID 2326646 swapped in MINIX: PID 15094 created MINIX: PID 2326647 swapped in MINIX: PID 15095 created MINIX: PID 2326648 swapped in MINIX: PID 15096 created MINIX: PID 2326649 swapped in MINIX: PID 15097 created MINIX: PID 2326650 swapped in MINIX: PID 15098 created MINIX: PID 2326651 swapped in MINIX: PID 15093 exitedMINIX: PID 15099 created MINIX: PID 2326652 swapped in MINIX: PID 15094 exitedMINIX: PID 15100 created MINIX: PID 2326653 swapped in MINIX: PID 15095 exitedMINIX: PID 15096 exitedMINIX: PID 15097 exitedMINIX: PID 15098 exitedMINIX: PID 15100 exitedMINIX: PID 15099 exited# </pre> <p>As it could be seen in the image above processes are created in the order 15093, 15094, 15095, 15096, 15097, 15098 and are excited in the same order. This is because FIFO policy schedules the first ready process first.</p>

