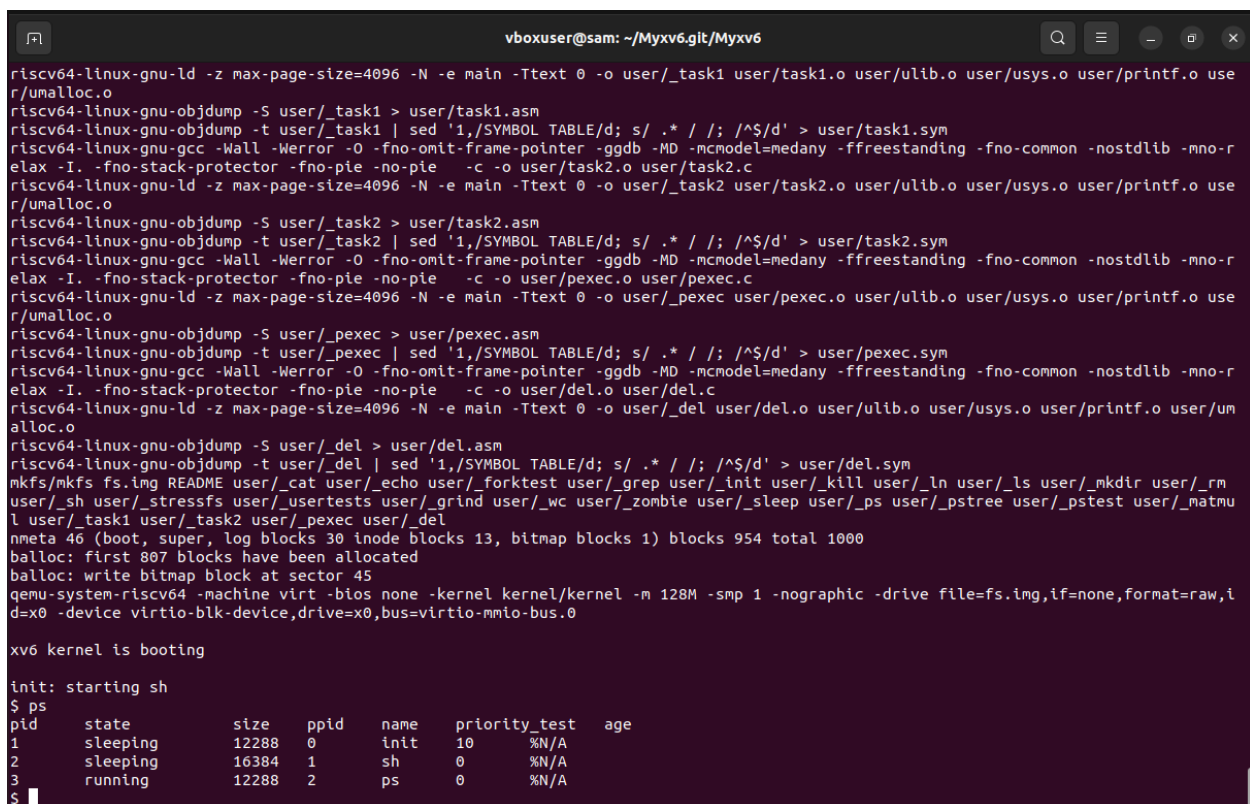


Priority-based Scheduler for xv6

Task 1. Modify the provided `ps` command to print the priority of each process.

In order to the function `getprocs()` to retrieve and print the priority field I had to add the following line of code to the helper function `procinfo()` : `procinfo.priority = p->priority;`. For this task I also added the priority field to `struct proc` in `pstat.h`. For the implementation of `get/set priority` system calls, I added the code for their functions. In `ps.c` I simply called the priority and printed it.



```
vboxuser@sam: ~/Myxv6.git/Myxv6
riscv64-linux-gnu-ld -z max-page-size=4096 -N -e main -Ttext 0 -o user/_task1 user/task1.o user/ulib.o user/usys.o user/printf.o user/_umalloc.o
riscv64-linux-gnu-objdump -S user/_task1 > user/task1.asm
riscv64-linux-gnu-objdump -t user/_task1 | sed '1,/SYMBOL TABLE/d; s/ .* / /; /^$/d' > user/task1.sym
riscv64-linux-gnu-gcc -Wall -Werror -O -fno-omit-frame-pointer -ggdb -MD -mcmodel=medany -ffreestanding -fno-common -nostdlib -mno-relax -I. -fno-stack-protector -fno-pie -no-pie -c -o user/task2.o user/task2.c
riscv64-linux-gnu-ld -z max-page-size=4096 -N -e main -Ttext 0 -o user/_task2 user/task2.o user/ulib.o user/usys.o user/printf.o user/_umalloc.o
riscv64-linux-gnu-objdump -S user/_task2 > user/task2.asm
riscv64-linux-gnu-objdump -t user/_task2 | sed '1,/SYMBOL TABLE/d; s/ .* / /; /^$/d' > user/task2.sym
riscv64-linux-gnu-gcc -Wall -Werror -O -fno-omit-frame-pointer -ggdb -MD -mcmodel=medany -ffreestanding -fno-common -nostdlib -mno-relax -I. -fno-stack-protector -fno-pie -no-pie -c -o user/pexec.o user/pexec.c
riscv64-linux-gnu-ld -z max-page-size=4096 -N -e main -Ttext 0 -o user/_pexec user/pexec.o user/ulib.o user/usys.o user/printf.o user/_umalloc.o
riscv64-linux-gnu-objdump -S user/_pexec > user/pexec.asm
riscv64-linux-gnu-objdump -t user/_pexec | sed '1,/SYMBOL TABLE/d; s/ .* / /; /^$/d' > user/pexec.sym
riscv64-linux-gnu-gcc -Wall -Werror -O -fno-omit-frame-pointer -ggdb -MD -mcmodel=medany -ffreestanding -fno-common -nostdlib -mno-relax -I. -fno-stack-protector -fno-pie -no-pie -c -o user/del.o user/del.c
riscv64-linux-gnu-ld -z max-page-size=4096 -N -e main -Ttext 0 -o user/_del user/del.o user/ulib.o user/usys.o user/printf.o user/_umalloc.o
riscv64-linux-gnu-objdump -S user/_del > user/del.asm
riscv64-linux-gnu-objdump -t user/_del | sed '1,/SYMBOL TABLE/d; s/ .* / /; /^$/d' > user/del.sym
mkfs/mkfs fs.img README user/_cat user/_echo user/_forktest user/_grep user/_init user/_ln user/_ls user/_mkdir user/_rm user/_sh user/_stressfs user/_usertests user/_grind user/_wc user/_zombie user/_sleep user/_ps user/_pstree user/_ptest user/_matnu l user/_task1 user/_task2 user/_pexec user/_del
nmtest 46 (boot, super, log blocks 30 inode blocks 13, bitmap blocks 1) blocks 954 total 1000
ballocc: first 807 blocks have been allocated
ballocc: write bitmap block at sector 45
qemu-system-riscv64 -machine virt -bios none -kernel kernel/kernel -m 128M -smp 1 -nographic -drive file=fs.img,if=none,format=raw,id=x0 -device virtio-blk-device,drive=x0,bus=virtio-mmio-bus.0

xv6 kernel is booting

init: starting sh
$ ps
pid  state  size  ppid  name  priority_test  age
1    sleeping  12288  0    init  10           %N/A
2    sleeping  16384  1    sh    0           %N/A
3    running   12288  2    ps    0           %N/A
$
```

Figure 1: Results of modified `ps`

Task 2. Add a readytime field to `struct proc`, initialize it correctly, and modify `ps` to print a process's age.

The way I'm calculating the process's age is using the following line of code on `ps.c` : `int age = current_time - uproc[i].readytime;`. The `current_time` is capturing the current system time by calling `uptime()` function. The `readytime` field was added to `struct proc` and the following line of

code was added inside *wakeup()* function in *proc.c* : *p->readytime = ticks;*. I modified *ps.c* file to print the age when if the process state is RUNNABLE.

```

vboxuser@sam: ~/Myxv6.git/Myxv6
ballocc: first 807 blocks have been allocated
ballocc: write bitmap block at sector 45
qemu-system-riscv64 -machine virt -bios none -kernel kernel/kernel -m 128M -smp 1 -nographic -drive file=fs.img,if=none,format=raw,i
d=x0 -device virtio-blk-device,drive=x0,bus=virtio-mmio-bus.0

xv6 kernel is booting

init: starting sh
$ pexec 5 matmul 5 &; matmul 10 &
$ pexec 10 ps
pid  state      size    ppid    name     priority_test  age
1    sleeping    12288   0       init     10             0
2    sleeping    16384   1       sh       0              0
7    runnable    12288   5       matmul   0              54
6    runnable    12288   1       matmul   0              54
5    sleeping    12288   1       pexec    0              0
8    sleeping    12288   2       pexec    0              0
9    running     12288   8       ps       0              0
$ pexec 10 ps
pid  state      size    ppid    name     priority_test  age
1    sleeping    12288   0       init     10             0
2    sleeping    16384   1       sh       0              0
7    runnable    12288   5       matmul   0              130
6    runnable    12288   1       matmul   0              130
5    sleeping    12288   1       pexec    0              0
10   sleeping    12288   2       pexec    0              0
11   running     12288   10      ps       0              0
$ pexec 10 p
$ s
exec s failed
$ peTime: 200 ticks
pexec 10 ps
pid  state      size    ppid    name     priority_test  age
1    sleeping    12288   0       init     10             0
2    sleeping    16384   1       sh       0              0
15   sleeping    12288   2       pexec    0              0
6    runnable    12288   1       matmul   0              254
16   running     12288   15      ps       0              0

```

Figure 2: Results of running *ps* command. I ran *pexec 5 matmul 5 &; matmul 10 &* then *pexec 10 ps* to get runnable processes.

Task 3. Implement a priority-based scheduler.

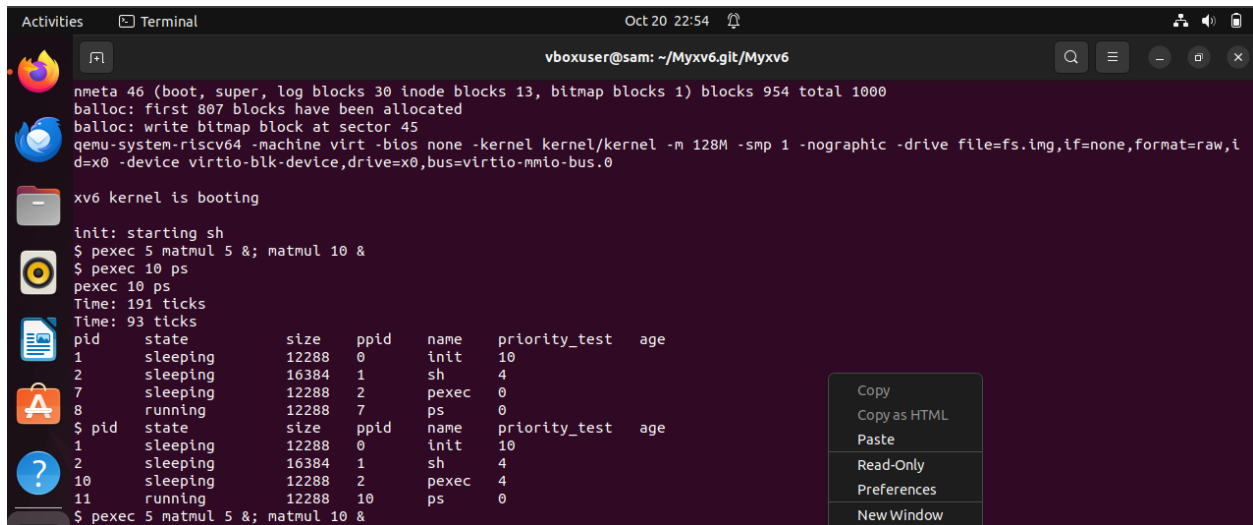
Param.h was changed and the constants of Round robin scheduler which was the scheduler xv6 uses and the new priority based scheduler were added. The purpose of this change is to have an option to switch between policies.

Proc.c inside the *scheduler* function was modified to add the priority-based scheduler inside the scheduler function. The purpose of the changes is to check what the current policy is and to change how the process will run next depending on the policy.

The changes have been pushed to the repo.

I learned about the different scheduling policies that can be added to the scheduler function and how they change the scheduling of the processes. With this I'm learning more about how processes work and how I can manage them depending on the situation.

One of the difficulties I faced was getting panic release message when creating the priority scheduling code. I modified the code and was getting no error or warning when running qemu but as soon as xv6 started I got the panic message. To fix it I had to research on what and why I was getting the panic message and after doing some research I found out that I was acquiring and releasing locks incorrectly.



```
nvmeta 46 (boot, super, log blocks 30 inode blocks 13, bitmap blocks 1) blocks 954 total 1000
ballocc: first 807 blocks have been allocated
ballocc: write bitmap block at sector 45
qemu-system-riscv64 -machine virt -bios none -kernel kernel/kernel -m 128M -smp 1 -nographic -drive file=fs.img,if=none,format=raw,i
d=x0 -device virtio-blk-device,drive=x0,bus=virtio-mmio-bus.0

xv6 kernel is booting

init: starting sh
$ pexec 5 matmul 5 & matmul 10 &
$ pexec 10 ps
pexec 10 ps
Time: 191 ticks
Time: 93 ticks
pid      state      size    ppid    name     priority_test  age
1        sleeping  12288   0       init     10             0
2        sleeping  16384   1       sh       4              0
7        sleeping  12288   2       pexec    0              0
8        running   12288   7       ps       0              0
$ pid      state      size    ppid    name     priority_test  age
1        sleeping  12288   0       init     10             0
2        sleeping  16384   1       sh       4              0
10       sleeping  12288   2       pexec    4              0
11       running   12288   10      ps       0              0
$ pexec 5 matmul 5 & matmul 10 &
```

Figure 3: Testing pexec and ps to test priority-based scheduler changes.

Task 4. Add aging to your priority based scheduler.

The aging policy I added to the priority based scheduler was that the longer the process is (age is higher) it will be the first one to run.

I was able to learn how the age of a process can be obtained and calculated in xv6 and how there are policies that work depending on the age of each process.

The difficulties I had were not knowing when to start recording or saving the age of the process.

I was able to solve this by reading and researching more about aging policies. Another issue I had was implementing this in the priority schedule. I was getting a lot of errors and couldn't figure out how to make implement this properly.

```
Activities Terminal Oct 20 22:56 vboxuser@sam: ~/Myxv6.git/Myxv6

riscv64-linux-gnu-objdump -S user/_del > user/del.asm
riscv64-linux-gnu-objdump -t user/_del | sed '1,/SYMBOL TABLE/d; s/ .* / /; /^$/d' > user/del.sym
mkfs/mkfs fs.img README user/_cat user/_echo user/_forktest user/_grep user/_init user/_kill user/_ln user/_ls user/_mkdir user/_rm
user/_sh user/_stressfs user/_usertests user/_grind user/_wc user/_zombie user/_sleep user/_ps user/_pstree user/_ptest user/_matmul
l user/_task1 user/_task2 user/_pexec user/_del
nmeta 46 (boot, super, log blocks 30 inode blocks 13, bitmap blocks 1) blocks 954 total 1000
ballocc: first 807 blocks have been allocated
ballocc: write bitmap block at sector 45
qemu-system-riscv64 -machine virt -bios none -kernel kernel/kernel -m 128M -smp 1 -nographic -drive file=fs.img,if=none,format=raw,i
d=x0 -device virtio-blk-device,drive=x0,bus=virtio-mmio-bus.0

xv6 kernel is booting

init: starting sh
$ ps
  pid  state      size  ppid  name  priority_test  age
  1    sleeping  12288   0    init   10             0
  2    sleeping  16384   1     sh    4             0
  3    running   12288   2     ps    0             0
$ pexec 5 matmul 5 &; matmul 10 &
$ pexec 10 ps
  pid  state      size  ppid  name  priority_test  age
  1    sleeping  12288   0    init   10             0
  2    sleeping  16384   1     sh    4             0
  8    runnable  12288   6    matmul  4             114
  7    runnable  12288   1    matmul  0             114
  6    sleeping  12288   1    pexec   4             0
  9    sleeping  12288   2    pexec   0             0
 10    running   12288   9     ps    0             0
$ pexec 10 ps
  pid  state      size  ppid  name  priority_test  age
  1    sleeping  12288   0    init   10             0
  2    sleeping  16384   1     sh    4             0
  8    runnable  12288   6    matmul  4             216
  7    runnable  12288   1    matmul  0             216
  6    sleeping  12288   1    pexec   4             0
 11    sleeping  12288   2    pexec   0             0
 12    running   12288  11     ps    0             0
$ Time: 192 ticks
```

Figure 4: Results after implementing aging.

These commands can improve the response time for interactive jobs because you can manually set or get the priority for any current process being used. You are able to manage processes and control any resource allocation.

I tried implementing the commands but I didn't have enough time to fix the errors the files are in /user