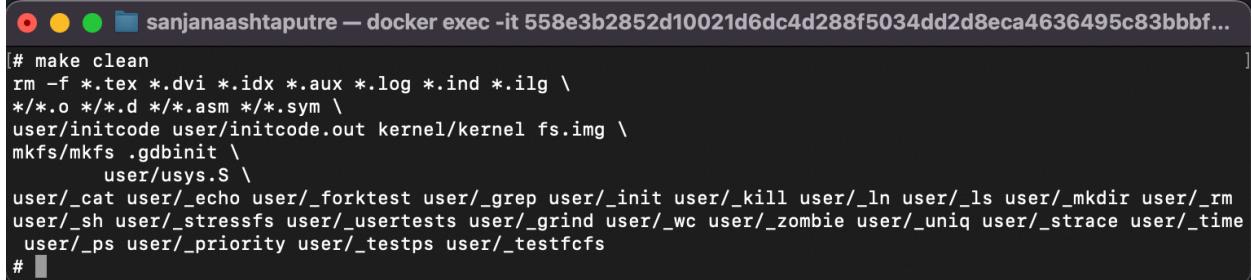


Priority-based Scheduling

Implemented in XV6-RISCV in DOCKER (in Mac OS - M1)

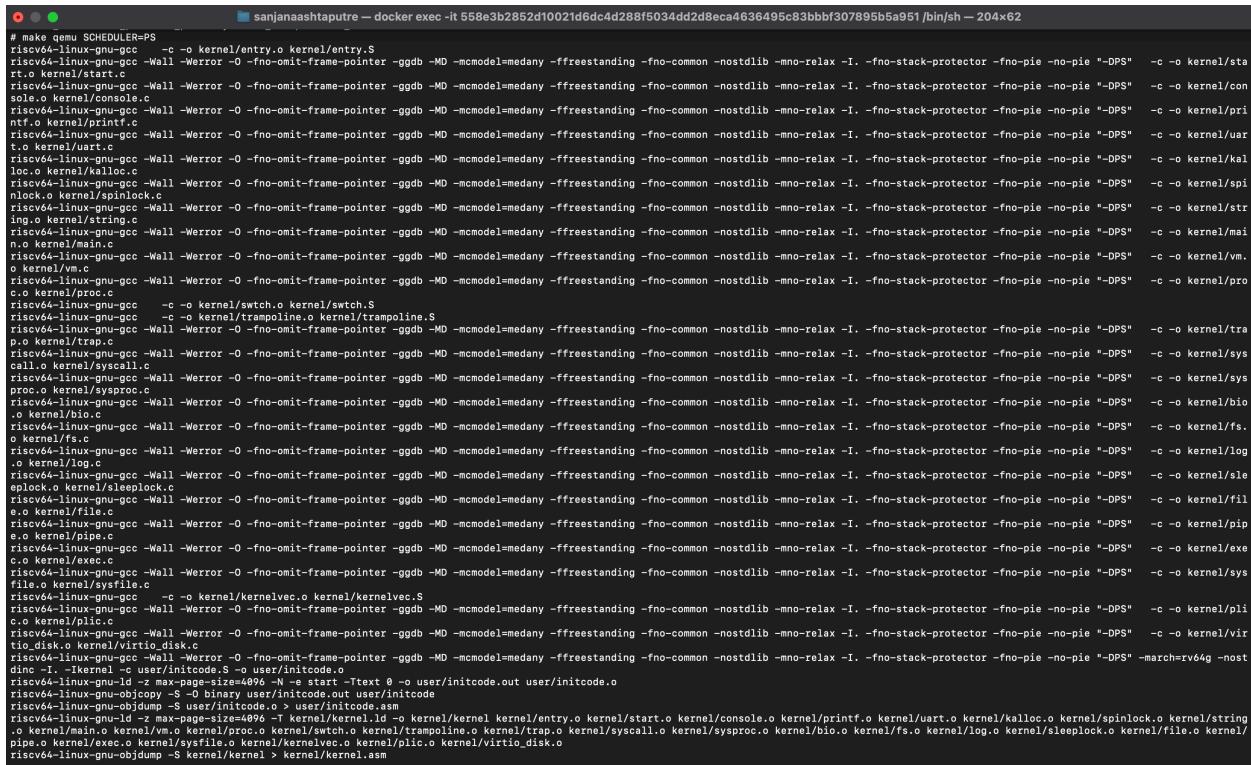
Steps to run the code and its output:

make clean



```
# make clean
rm -f *.tex *.dvi *.idx *.aux *.log *.ind *.ilg \
/*!*.o */*.d */*.asm */*.sym \
user/initcode user/initcode.out kernel/kernel fs.img \
mkfs/mkfs .gdbinit \
user/usys.S \
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/_uniq user/_strace user/_time \
user/_ps user/_priority user/_testps user/_testfcfs
#
```

make qemu SCHEDULER=PS



```
# make qemu SCHEDULER=PS
riscv64-linux-gnu-gcc -c -o kernel/entry.o kernel/entry.S
riscv64-linux-gnu-gcc -Wall -Werror -O -fno-omit-frame-pointer -ggdb -MD -mcpu=medany -ffreestanding -fno-common -nostdlib -mno-relax -I. -fno-stack-protector -fno-pie -no-pie "-DPS" -c -o kernel/start.c
riscv64-linux-gnu-gcc -Wall -Werror -O -fno-omit-frame-pointer -ggdb -MD -mcpu=medany -ffreestanding -fno-common -nostdlib -mno-relax -I. -fno-stack-protector -fno-pie -no-pie "-DPS" -c -o kernel/console.c
riscv64-linux-gnu-gcc -Wall -Werror -O -fno-omit-frame-pointer -ggdb -MD -mcpu=medany -ffreestanding -fno-common -nostdlib -mno-relax -I. -fno-stack-protector -fno-pie -no-pie "-DPS" -c -o kernel/print.c
riscv64-linux-gnu-gcc -Wall -Werror -O -fno-omit-frame-pointer -ggdb -MD -mcpu=medany -ffreestanding -fno-common -nostdlib -mno-relax -I. -fno-stack-protector -fno-pie -no-pie "-DPS" -c -o kernel/usrt.c
riscv64-linux-gnu-gcc -Wall -Werror -O -fno-omit-frame-pointer -ggdb -MD -mcpu=medany -ffreestanding -fno-common -nostdlib -mno-relax -I. -fno-stack-protector -fno-pie -no-pie "-DPS" -c -o kernel/kalloc.c
riscv64-linux-gnu-gcc -Wall -Werror -O -fno-omit-frame-pointer -ggdb -MD -mcpu=medany -ffreestanding -fno-common -nostdlib -mno-relax -I. -fno-stack-protector -fno-pie -no-pie "-DPS" -c -o kernel/spinlock.c
riscv64-linux-gnu-gcc -Wall -Werror -O -fno-omit-frame-pointer -ggdb -MD -mcpu=medany -ffreestanding -fno-common -nostdlib -mno-relax -I. -fno-stack-protector -fno-pie -no-pie "-DPS" -c -o kernel/string.c
riscv64-linux-gnu-gcc -Wall -Werror -O -fno-omit-frame-pointer -ggdb -MD -mcpu=medany -ffreestanding -fno-common -nostdlib -mno-relax -I. -fno-stack-protector -fno-pie -no-pie "-DPS" -c -o kernel/main.c
riscv64-linux-gnu-gcc -Wall -Werror -O -fno-omit-frame-pointer -ggdb -MD -mcpu=medany -ffreestanding -fno-common -nostdlib -mno-relax -I. -fno-stack-protector -fno-pie -no-pie "-DPS" -c -o kernel/vmem.c
riscv64-linux-gnu-gcc -Wall -Werror -O -fno-omit-frame-pointer -ggdb -MD -mcpu=medany -ffreestanding -fno-common -nostdlib -mno-relax -I. -fno-stack-protector -fno-pie -no-pie "-DPS" -c -o kernel/proc.c
riscv64-linux-gnu-gcc -c -o kernel/switch.o kernel/switch.S
riscv64-linux-gnu-gcc -c -o kernel/trampoline.o kernel/trampoline.S
riscv64-linux-gnu-gcc -Wall -Werror -O -fno-omit-frame-pointer -ggdb -MD -mcpu=medany -ffreestanding -fno-common -nostdlib -mno-relax -I. -fno-stack-protector -fno-pie -no-pie "-DPS" -c -o kernel/trap.c
riscv64-linux-gnu-gcc -Wall -Werror -O -fno-omit-frame-pointer -ggdb -MD -mcpu=medany -ffreestanding -fno-common -nostdlib -mno-relax -I. -fno-stack-protector -fno-pie -no-pie "-DPS" -c -o kernel/syscall.c
riscv64-linux-gnu-gcc -Wall -Werror -O -fno-omit-frame-pointer -ggdb -MD -mcpu=medany -ffreestanding -fno-common -nostdlib -mno-relax -I. -fno-stack-protector -fno-pie -no-pie "-DPS" -c -o kernel/sysproc.c
riscv64-linux-gnu-gcc -Wall -Werror -O -fno-omit-frame-pointer -ggdb -MD -mcpu=medany -ffreestanding -fno-common -nostdlib -mno-relax -I. -fno-stack-protector -fno-pie -no-pie "-DPS" -c -o kernel/sleeplock.c
riscv64-linux-gnu-gcc -Wall -Werror -O -fno-omit-frame-pointer -ggdb -MD -mcpu=medany -ffreestanding -fno-common -nostdlib -mno-relax -I. -fno-stack-protector -fno-pie -no-pie "-DPS" -c -o kernel/bio.c
riscv64-linux-gnu-gcc -Wall -Werror -O -fno-omit-frame-pointer -ggdb -MD -mcpu=medany -ffreestanding -fno-common -nostdlib -mno-relax -I. -fno-stack-protector -fno-pie -no-pie "-DPS" -c -o kernel/fs.c
riscv64-linux-gnu-gcc -Wall -Werror -O -fno-omit-frame-pointer -ggdb -MD -mcpu=medany -ffreestanding -fno-common -nostdlib -mno-relax -I. -fno-stack-protector -fno-pie -no-pie "-DPS" -c -o kernel/log.c
riscv64-linux-gnu-gcc -Wall -Werror -O -fno-omit-frame-pointer -ggdb -MD -mcpu=medany -ffreestanding -fno-common -nostdlib -mno-relax -I. -fno-stack-protector -fno-pie -no-pie "-DPS" -c -o kernel/sles.c
riscv64-linux-gnu-gcc -Wall -Werror -O -fno-omit-frame-pointer -ggdb -MD -mcpu=medany -ffreestanding -fno-common -nostdlib -mno-relax -I. -fno-stack-protector -fno-pie -no-pie "-DPS" -c -o kernel/file.c
riscv64-linux-gnu-gcc -Wall -Werror -O -fno-omit-frame-pointer -ggdb -MD -mcpu=medany -ffreestanding -fno-common -nostdlib -mno-relax -I. -fno-stack-protector -fno-pie -no-pie "-DPS" -c -o kernel/pipe.c
riscv64-linux-gnu-gcc -Wall -Werror -O -fno-omit-frame-pointer -ggdb -MD -mcpu=medany -ffreestanding -fno-common -nostdlib -mno-relax -I. -fno-stack-protector -fno-pie -no-pie "-DPS" -c -o kernel/exec.c
riscv64-linux-gnu-gcc -Wall -Werror -O -fno-omit-frame-pointer -ggdb -MD -mcpu=medany -ffreestanding -fno-common -nostdlib -mno-relax -I. -fno-stack-protector -fno-pie -no-pie "-DPS" -c -o kernel/sysfile.c
riscv64-linux-gnu-gcc -c -o kernel/kernelvec.o kernel/kernelvec.S
riscv64-linux-gnu-gcc -Wall -Werror -O -fno-omit-frame-pointer -ggdb -MD -mcpu=medany -ffreestanding -fno-common -nostdlib -mno-relax -I. -fno-stack-protector -fno-pie -no-pie "-DPS" -c -o kernel/plic.c
riscv64-linux-gnu-gcc -Wall -Werror -O -fno-omit-frame-pointer -ggdb -MD -mcpu=medany -ffreestanding -fno-common -nostdlib -mno-relax -I. -fno-stack-protector -fno-pie -no-pie "-DPS" -c -o kernel/virtio-disk.c
riscv64-linux-gnu-gcc -Wall -Werror -O -fno-omit-frame-pointer -ggdb -MD -mcpu=medany -ffreestanding -fno-common -nostdlib -mno-relax -I. -fno-stack-protector -fno-pie -no-pie "-DPS" -c -o kernel/dinc.c
riscv64-linux-gnu-gcc -c user/initcode.o -o user/initcode
riscv64-linux-gnu-gcc -S -D max-page-size=4096 -N -e start -Ttext 0 -o user/initcode.out user/initcode.o
riscv64-linux-gnu-objdump -S -user/initcode.o > user/initcode.asm
riscv64-linux-gnu-gcc -z max-page-size=4096 -t kernel/kernel_id -o kernel/kernel.entry.o kernel/start.o kernel/console.o kernel/printf.o kernel/uart.o kernel/kalloc.o kernel/spinlock.o kernel/string.o kernel/main.o kernel/vm.o kernel/proc.o kernel/switch.o kernel/trampoline.o kernel/trap.o kernel/syscall.o kernel/bio.o kernel/log.o kernel/sleeplock.o kernel/file.o kernel/pipe.o kernel/exec.o kernel/sysfile.o kernel/kernelvec.o kernel/plic.o kernel/virtio_disk.o
riscv64-linux-gnu-objdump -S kernel/kernel > kernel/kernel.asm
```

```
user@wc.c
riscv64-linux-gnu-ld -z max-page-size=4096 -N -e main -Ttext 0 -o user/_wc.o user/_libc.o user/_usys.o user/_printf.o user/_umalloc.o
riscv64-linux-gnu-ld -b _start -Ttext 0 -o user/_wc.o user/_wc.asm
riscv64-linux-gnu-ld -b _start -Ttext 0 -o user/_wc | sed '1,/SYMBOL TABLE/d; s/.* /;/;$d' > user/_wc.sym
riscv64-linux-gnu-ld -z max-page-size=4096 -N -e main -Ttext 0 -o user/_zombie.o user/_zombie.o user/_libc.o user/_usys.o user/_printf.o user/_umalloc.o
riscv64-linux-gnu-ld -b _start -Ttext 0 -o user/_zombie > user/_zombie.asm
riscv64-linux-gnu-ld -b _start -Ttext 0 -o user/_zombie | sed '1,/SYMBOL TABLE/d; s/.* /;/;$d' > user/_zombie.sym
riscv64-linux-gnu-ld -z max-page-size=4096 -N -e main -TText 0 -o user/_uniq_user.o user/_uniq.o user/_libc.o user/_usys.o user/_printf.o user/_umalloc.o
riscv64-linux-gnu-ld -b _start -Ttext 0 -o user/_uniq_user > user/_uniq.asm
riscv64-linux-gnu-ld -b _start -Ttext 0 -o user/_uniq | sed '1,/SYMBOL TABLE/d; s/.* /;/;$d' > user/_uniq.sym
riscv64-linux-gnu-ld -z max-page-size=4096 -N -e main -TText 0 -o user/_strace.o user/_strace.o user/_libc.o user/_usys.o user/_printf.o user/_umalloc.o
riscv64-linux-gnu-ld -b _start -Ttext 0 -o user/_strace > user/_strace.asm
riscv64-linux-gnu-ld -b _start -Ttext 0 -o user/_strace | sed '1,/SYMBOL TABLE/d; s/.* /;/;$d' > user/_strace.sym
riscv64-linux-gnu-gcc -Wall -Werror -O -fno-omit-frame-pointer -ggdb -MD -mcmdel=medany -ffreestanding -fno-common -nostdlib -mno-relax -I. -fno-stack-protector -fno-pie -no-pie --DPS" -c -o user/_time.o
riscv64-linux-gnu-ld -z max-page-size=4096 -N -e main -TText 0 -o user/_time.o user/_libc.o user/_usys.o user/_printf.o user/_umalloc.o
riscv64-linux-gnu-ld -b _start -Ttext 0 -o user/_time > user/_time.asm
riscv64-linux-gnu-ld -b _start -Ttext 0 -o user/_time | sed '1,/SYMBOL TABLE/d; s/.* /;/;$d' > user/_time.sym
riscv64-linux-gnu-gcc -Wall -Werror -O -fno-omit-frame-pointer -ggdb -MD -mcmdel=medany -ffreestanding -fno-common -nostdlib -mno-relax -I. -fno-stack-protector -fno-pie -no-pie --DPS" -c -o user/_ps.o
riscv64-linux-gnu-ld -z max-page-size=4096 -N -e main -TText 0 -o user/_ps.o user/_ps.o user/_libc.o user/_usys.o user/_printf.o user/_umalloc.o
riscv64-linux-gnu-ld -b _start -Ttext 0 -o user/_ps > user/_ps.asm
riscv64-linux-gnu-ld -b _start -Ttext 0 -o user/_ps | sed '1,/SYMBOL TABLE/d; s/.* /;/;$d' > user/_ps.sym
riscv64-linux-gnu-gcc -Wall -Werror -O -fno-omit-frame-pointer -ggdb -MD -mcmdel=medany -ffreestanding -fno-common -nostdlib -mno-relax -I. -fno-stack-protector -fno-pie -no-pie --DPS" -c -o user/_prior.o
riscv64-linux-gnu-ld -z max-page-size=4096 -N -e main -TText 0 -o user/_priority.o user/_libc.o user/_usys.o user/_printf.o user/_umalloc.o
riscv64-linux-gnu-ld -b _start -Ttext 0 -o user/_priority > user/_priority.asm
riscv64-linux-gnu-ld -b _start -Ttext 0 -o user/_priority | sed '1,/SYMBOL TABLE/d; s/.* /;/;$d' > user/_priority.sym
riscv64-linux-gnu-gcc -Wall -Werror -O -fno-omit-frame-pointer -ggdb -MD -mcmdel=medany -ffreestanding -fno-common -nostdlib -mno-relax -I. -fno-stack-protector -fno-pie -no-pie --DPS" -c -o user/_testsp.o
riscv64-linux-gnu-ld -z max-page-size=4096 -N -e main -TText 0 -o user/_testsp.o user/_testsp.o user/_libc.o user/_usys.o user/_printf.o user/_umalloc.o
riscv64-linux-gnu-ld -b _start -Ttext 0 -o user/_testsp > user/_testsp.asm
riscv64-linux-gnu-ld -b _start -Ttext 0 -o user/_testsp | sed '1,/SYMBOL TABLE/d; s/.* /;/;$d' > user/_testsp.sym
riscv64-linux-gnu-gcc -Wall -Werror -O -fno-omit-frame-pointer -ggdb -MD -mcmdel=medany -ffreestanding -fno-common -nostdlib -mno-relax -I. -fno-stack-protector -fno-pie -no-pie --DPS" -c -o user/_testfcs.o
riscv64-linux-gnu-ld -z max-page-size=4096 -N -e main -TText 0 -o user/_testfcfs.o user/_testfcfs.o user/_libc.o user/_usys.o user/_printf.o user/_umalloc.o
riscv64-linux-gnu-ld -b _start -Ttext 0 -o user/_testfcfs > user/_testfcfs.asm
riscv64-linux-gnu-ld -b _start -Ttext 0 -o user/_testfcfs | sed '1,/SYMBOL TABLE/d; s/.* /;/;$d' > user/_testfcfs.sym
mkfs/mkfs fs.img README.md 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/_userstests user/_grind user/_wc user/_zombie user/_uniq_user/_strace user/_time user/_ps user/_priority user/_testsp user/_testfcfs
nmetas 44 (boot, super, log blocks 30 inode blocks 13, bitmap blocks 1) blocks 954 total 1000
bitmap first 774 blocks have been allocated
bitmap first 774 blocks have been allocated
mkfs/mkfs fs.img unifile.tst.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/_userstests user/_grind user/_wc user/_zombie user/_uniq_user/_strace user/_time user/_ps user/_priority user/_testsp user/_testfcfs
nmetas 44 (boot, super, log blocks 30 inode blocks 13, bitmap blocks 1) blocks 954 total 1000
balloc: first 774 blocks have been allocated
balloc: write bitmap block at sector 45
gemu-system-riscv64 -machine virt -bios none -kernel kernel/kernel -m 128M -smp 3 -nographic -drive file=fs.img,if=none,format=raw,id=x0 -device virtio-blk-device,drive=x0,bus=virtio-mmcio-bus.0
xv kernel is booting

hart 2 starting
init: starting sh
$ 
```

Input files:

```
[sanjanaashtaputre - docker exec -it 558e3b2852c]$ cat uniqfile.txt
I understand the Operating system.
I understand the Operating system.
I understand the Operating system.
I love to work on OS.
I love to work on OS.
Thanks xv6.
THANKS XV6.
$
```

Output:

<program file> <number_of_processes> <Input_file>

testps 4 uniqfile.txt

```
hart 1 starting
init: starting sh
$ testps 4 uniqfile.txt

Child PID 4 created

Uniq command is getting executed in kernel mode
I understand the Operating system.
I love to work on OS.
Thanks xv6.
THANKS XV6.

creation time : 1153
run time : 25
end time : 1181
wait time : 3
Turnaround Time : 28

Child PID 5 created
Uniq command is getting executed in user mode.
I understand the Operating system.
I love to work on OS.
Thanks xv6.
THANKS XV6.

creation time : 1181
run time : 27
end time : 1210
wait time : 2
Turnaround Time : 29

Child PID 6 created

Uniq command is getting executed in kernel mode
I understand the Operating system.
I love to work on OS.
Thanks xv6.
THANKS XV6.

creation time : 1210
run time : 27
end time : 1239
wait time : 2
Turnaround Time : 29

Child PID 7 created
Uniq command is getting executed in user mode.
I understand the Operating system.
I love to work on OS.
Thanks xv6.
THANKS XV6.

creation time : 1239
run time : 26
end time : 1269
wait time : 4
Turnaround Time : 30

Average run time 26, Average wait time 2
$
```

```
<program file> <number_of_processes> <-c/-d/-i> <Input_file>
```

```
testps 3 -c uniqfile.txt
```

```
sanjanaashtaputre — docker exec -it 558e3b2852d10021d6dc4d288f503$ testps 3 -c uniqfile.txt

Child PID 16 created

Uniq command is getting executed in kernel mode
3 I understand the Operating system.
2 I love to work on OS.
1 Thanks xv6.
1 THANKS XV6.

creation time : 3291
run time : 26
end time : 3319
wait time : 2
Turnaround Time : 28

Child PID 17 created
Uniq command is getting executed in user mode.
3 I understand the Operating system.
2 I love to work on OS.
1 Thanks xv6.
1 THANKS XV6.

creation time : 3319
run time : 27
end time : 3347
wait time : 1
Turnaround Time : 28

Child PID 18 created

Uniq command is getting executed in kernel mode
3 I understand the Operating system.
2 I love to work on OS.
1 Thanks xv6.
1 THANKS XV6.

creation time : 3348
run time : 26
end time : 3375
wait time : 1
Turnaround Time : 27

Average run time 26, Average wait time 1
$
```

testps 6 -i uniqfile.txt

```
sanjanaashtaputre — docker exec -it 558e3b2852d10021
$ testps 6 -i uniqfile.txt

Child PID 20 created

Uniq command is getting executed in kernel mode
I understand the Operating system.
I love to work on OS.
Thanks xv6.

creation time : 3952
run time : 24
end time : 3981
wait time : 5
Turnaround Time : 29

Child PID 21 created
Uniq command is getting executed in user mode.
I understand the Operating system.
I love to work on OS.
Thanks xv6.

creation time : 3981
run time : 24
end time : 4009
wait time : 4
Turnaround Time : 28

Child PID 22 created

Uniq command is getting executed in kernel mode
I understand the Operating system.
I love to work on OS.
Thanks xv6.

creation time : 4009
run time : 28
end time : 4038
wait time : 1
Turnaround Time : 29

Child PID 23 created
Uniq command is getting executed in user mode.
I understand the Operating system.
I love to work on OS.
Thanks xv6.

creation time : 4038
run time : 27
end time : 4068
wait time : 3
Turnaround Time : 30

Child PID 24 created

Uniq command is getting executed in kernel mode
I understand the Operating system.
I love to work on OS.
Thanks xv6.

creation time : 4068
run time : 27
end time : 4095
wait time : 0
Turnaround Time : 27

Child PID 25 created
Uniq command is getting executed in user mode.
I understand the Operating system.
I love to work on OS.
Thanks xv6.

creation time : 4096
run time : 29
end time : 4125
wait time : 0
Turnaround Time : 29

Average run time 26, Average wait time 2
$
```

```
testps 4 -d uniqfile.txt
```

```
$ testps 4 -d uniqfile.txt

Child PID 33 created

Uniq command is getting executed in kernel mode
I understand the Operating system.
I love to work on OS.

creation time : 7156
run time : 28
end time : 7186
wait time : 2
Turnaround Time : 30

Child PID 34 created
Uniq command is getting executed in user mode.
I understand the Operating system.
I love to work on OS.

creation time : 7186
run time : 27
end time : 7214
wait time : 1
Turnaround Time : 28

Child PID 35 created

Uniq command is getting executed in kernel mode
I understand the Operating system.
I love to work on OS.

creation time : 7214
run time : 23
end time : 7243
wait time : 6
Turnaround Time : 29

Child PID 36 created
Uniq command is getting executed in user mode.
I understand the Operating system.
I love to work on OS.

creation time : 7243
run time : 28
end time : 7272
wait time : 1
Turnaround Time : 29

Average run time 26, Average wait time 2
$
```

Code logic:

While running *make qemu*, Scheduler=PS should be mentioned. It is added to the Makefile accordingly.

File: **testps.c**

This file is created in the user directory of xv6. Here, a 'p' new processes are created using *fork()*. A dummy for loop is added just to consume run time with *uniq* function. Uniq command

is called both in user and kernel mode, once the process is completed, the statistics of the process are displayed. After running all the processes, the average wait and run time is calculated and displayed at the end. uniq_f is the user function containing the uniq functionality. The priority can be changed using the ‘priority’ command on the command prompt with process pid and priority.

```
cv-scheduling/user/testps.c
```

```

C
if(a!='-'){
    fdk = open(argv[i], 0);
    if (fdk < 0) {
        printf("uniq: cannot open \n");
    }
}
if(a=='-'){
    argv[i]++;
    var_icd = *argv[i];
    f=1;
}
if(f==0){
    if(k%2 == 0){
        while((n2 = read(fd़, buffer, sizeof(buffer))) > 0 )
        {
            uniq(n2, buffer, var_icd);
        }
        if (n2 < 0) {
            printf("Error: Unable to read file\n");
        }
    }
    else{
        uniq_f(fd़,var_icd);
    }
    close(fd़);
}
f=0;
}
exit(0);
}
else if ( pid < 0 ) {
    printf("%d failed in fork!\n", getpid());
}
else if (pid > 0) {
    // parent
    if(waitx(0,&rt,&wt) >= 0) {
        rt += rt;
        twt += wt;
    }
}
printf("\nAverage run time %d, Average wait time %d\n", rt / n, twt / n);
exit(0);

```

File: priority.c

In order to change priority of any process, a program is written in priority.c file where, the pid and priority of a process are taken as input and passed to pspr() function which replaces the priority of the process with the new priority.

```
/xv6-riscv-scheduling/user/priority.c
```

```

C
-----
2 #include "kernel/stat.h"
3 #include "user.h"
4
5 int main(int argc, char *argv[]) {
6     int priority, pid;
7     if(argc < 3){
8         printf("Usage: nice pid priority\n");
9         exit(1);
10    }
11    pid = atoi(argv[1]);
12    priority = atoi(argv[2]);
13    if (priority < 0 || priority > 20){
14        printf("Invalid priority (0-20)!\n");
15        exit(1);
16    }
17    pspr(pid, priority);
18    exit(0);
19 }
20

```

File: **defs.h**

The functions are declared here, like File:

```
int pspr(int pid, int priority);
```

File: exec.c

Here the child process priority is defaulted to 2 as it should have higher priority than the parent process that is defaulted to 10.

File: proc.c

In allocproc(), parent process priority is defaulted to 10 and all the other process fields are initialized.

In exit(), the process end time is obtained.

In scheduler(), the changes are made to the default round robin scheduling as shown below. Here, the process with higher priority is picked from RUNNABLE state and made to run with RUNNING state. The higher value means the priority is low. The scheduling time is incremented.

/xv6-riscv-scheduling/kernel/proc.c	C	/xv6-riscv-scheduling/kernel/proc.c	C
469 #ifdef PS	487 // Switch to chosen process. It is the process's job		
470	488 // to release its lock and then reacquire it		
471 struct proc *P2, *pl;	489 // before jumping back to us.		
472	490 p->state = RUNNING;		
473 for(p = proc; p < &proc[NPROC]; p++) {	491 c->proc = p;		
474 acquire(&p->lock);	492		
475 if(p->state == RUNNABLE) {	493 p->numScheduled++;		
476	494 swtch(&c->context, &p->context);		
477 P2 = p;	495		
478 //high priority for large values	496 // Process is done running for now.		
479 for(pl = proc; pl < &proc[NPROC]; pl++) {	497 // It should have changed its p->state before coming back.		
480 if(pl->state != RUNNABLE)	498		
481 continue;	499 c->proc = 0;		
482 if(P2->priority > pl->priority)	500		
483 P2 = pl;	501 release(&p->lock);		
484 }	502		
485 p = P2;	503 #else		
486	504		

`updateTime()` function is added to the file where the run time and sleep time of the process is updated.

`waitx()` function is added to the file, where total run time, turnaround time, wait time are calculated.

`pspr()` function is added which takes PID and priority as input and changes the current priority of the process to the new given priority.

```
995 int  
996 pspri(int pid, int priority)  
997 {  
998     struct proc *p;  
999     acquire(&wait_lock);  
1000    for(p = proc; p < &proc[NPROC]; p++){  
1001        if(p->pid == pid){  
1002            p->priority = priority;  
1003            break;  
1004        }  
1005    }  
1006    release(&wait_lock);  
1007    return pid;  
1008 }  
1009  
1010 void  
1011 updateTime()  
1012 {  
1013     struct proc *p;  
1014     for (p = proc; p < &proc[NPROC]; p++)  
1015     {  
1016         acquire(&p->lock);  
1017         if (p->state == RUNNING)  
1018         {  
1019             p->runTime++;  
1020             p->totalRuntime++;  
1021         }  
1022         if (p->state == SLEEPING)  
1023             p->sleepTime++;  
1024     }  
1025     release(&p->lock);  
1026 }
```

File: proc.h

All the process fields are declared here. Fields for priority, run time, end time etc.

File: syscall.c

All the function definitions are declared here like:

```
extern uint64 sys_pspr(void);
```

```
[SYS_pspr] sys_pspr,
```

File: syscall.h

All the system call numbers for the system calls are assigned here, like:

```
#define SYS_pspr 26
```

File: sysproc.c

```
257  uint64
258  sys_pspr(void)
259 v {
260     int pid, pr;
261     argint(0, &pid);
262     argint(1, &pr);
263
264     return pspr(pid, pr);
265 }
266
```

File: user.h

The system call function is declared like int pspr(int pid, int priority);

File: usys.pl

The system call entry is declared like:entry("pspr");