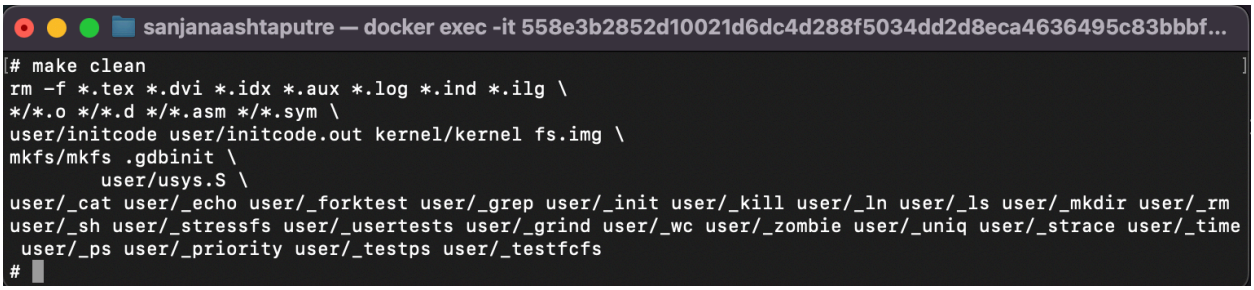
Priority-based Scheduling

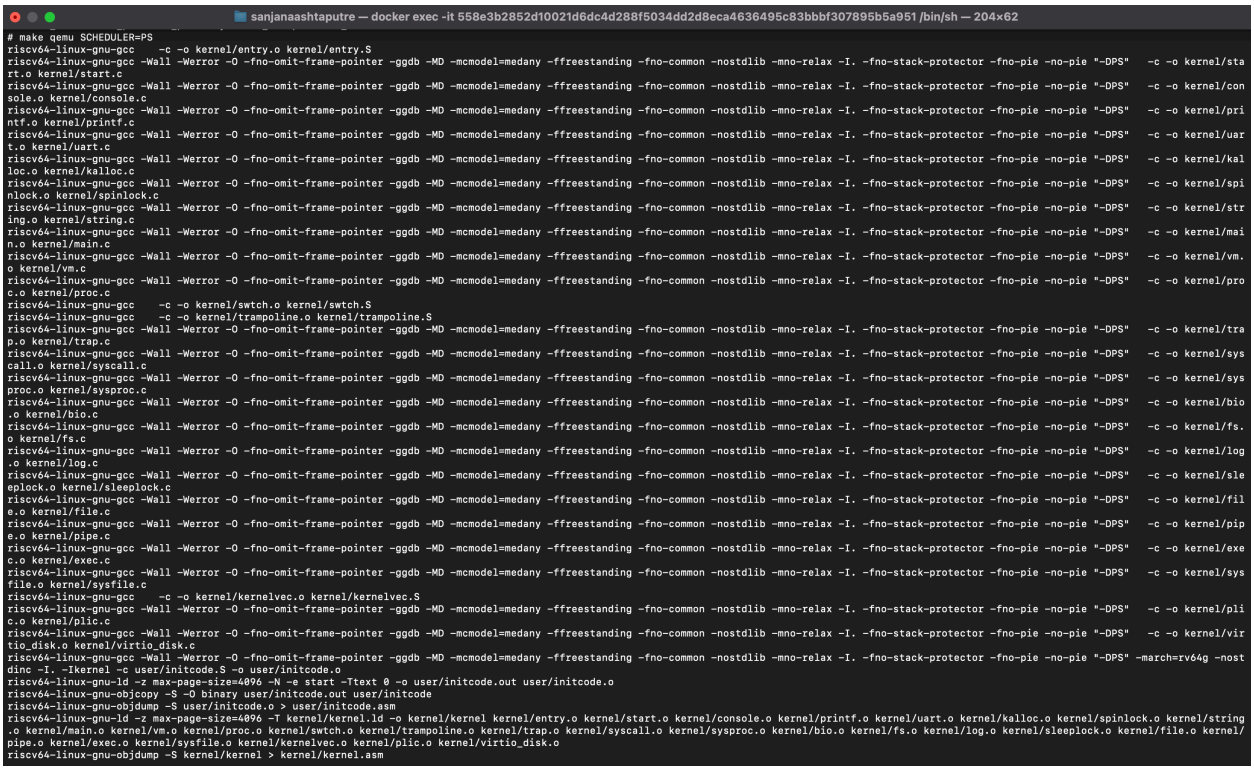
Implemented in XV6-RISCV in DOCKER (in Mac OS - M1)

Steps to run the code and its output:

make clean



make qemu SCHEDULER=PS

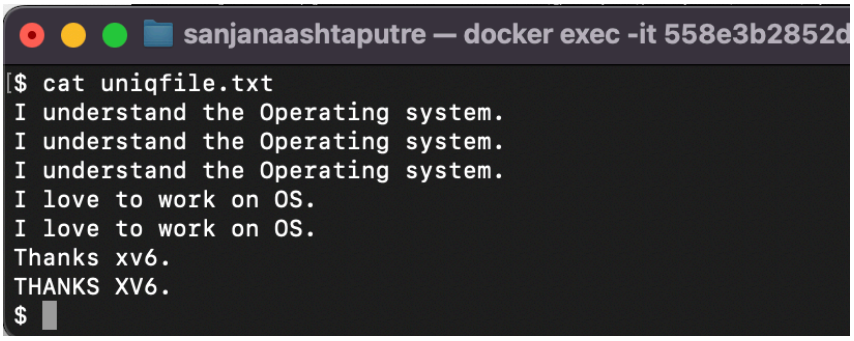


```
user/wc.c
riscv64-linux-gnu-ld -z max-page-size=4096 -N -e main -Ttext 0 -o user/_wc user/wc.o user/ulib.o user/usys.o user/printf.o user/umalloc.o
riscv64-linux-gnu-objdump -S user/_wc > user/wc.asm
riscv64-linux-gnu-objdump -t user/_wc | sed '1,/SYMBOL TABLE/d; s/ .* / /; /^$/d' > user/wc.sym
riscv64-linux-gnu-gcc -Wall -Werror -O -fno-omit-frame-pointer -ggdb -MD -mcmodel=medany -ffreestanding -fno-common -nostdlib -mno-relax -I. -fno-stack-protector -fno-pie -no-pie "-DPS"   -c -o user/zombie.o user/zombie.c
riscv64-linux-gnu-ld -z max-page-size=4096 -N -e main -Ttext 0 -o user/_zombie user/zombie.o user/ulib.o user/usys.o user/printf.o user/umalloc.o
riscv64-linux-gnu-objdump -S user/_zombie > user/zombie.asm
riscv64-linux-gnu-objdump -t user/_zombie | sed '1,/SYMBOL TABLE/d; s/ .* / /; /^$/d' > user/zombie.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 "-DPS"   -c -o user/uniq.o user/uniq.c
riscv64-linux-gnu-ld -z max-page-size=4096 -N -e main -Ttext 0 -o user/_uniq user/uniq.o user/ulib.o user/usys.o user/printf.o user/umalloc.o
riscv64-linux-gnu-objdump -S user/_uniq > user/uniq.asm
riscv64-linux-gnu-objdump -t user/_uniq | sed '1,/SYMBOL TABLE/d; s/ .* / /; /^$/d' > user/uniq.sym
riscv64-linux-gnu-gcc -Wall -Werror -O -fno-omit-frame-pointer -ggdb -MD -mcmodel=medany -ffreestanding -fno-common -nostdlib -mno-relax -I. -fno-stack-protector -fno-pie -no-pie "-DPS"   -c -o user/strace.o user/strace.c
riscv64-linux-gnu-ld -z max-page-size=4096 -N -e main -Ttext 0 -o user/_strace user/strace.o user/ulib.o user/usys.o user/printf.o user/umalloc.o
riscv64-linux-gnu-objdump -S user/_strace > user/strace.asm
riscv64-linux-gnu-objdump -t user/_strace | sed '1,/SYMBOL TABLE/d; s/ .* / /; /^$/d' > user/strace.sym
riscv64-linux-gnu-gcc -Wall -Werror -O -fno-omit-frame-pointer -ggdb -MD -mcmodel=medany -ffreestanding -fno-common -nostdlib -mno-relax -I. -fno-stack-protector -fno-pie -no-pie "-DPS"   -c -o user/time.o user/time.c
riscv64-linux-gnu-ld -z max-page-size=4096 -N -e main -Ttext 0 -o user/_time user/time.o user/ulib.o user/usys.o user/printf.o user/umalloc.o
riscv64-linux-gnu-objdump -S user/_time > user/time.asm
riscv64-linux-gnu-objdump -t user/_time | sed '1,/SYMBOL TABLE/d; s/ .* / /; /^$/d' > user/time.sym
riscv64-linux-gnu-gcc -Wall -Werror -O -fno-omit-frame-pointer -ggdb -MD -mcmodel=medany -ffreestanding -fno-common -nostdlib -mno-relax -I. -fno-stack-protector -fno-pie -no-pie "-DPS"   -c -o user/ps.o user/ps.c
riscv64-linux-gnu-ld -z max-page-size=4096 -N -e main -Ttext 0 -o user/_ps user/ps.o user/ulib.o user/usys.o user/printf.o user/umalloc.o
riscv64-linux-gnu-objdump -S user/_ps > user/ps.asm
riscv64-linux-gnu-objdump -t user/_ps | sed '1,/SYMBOL TABLE/d; s/ .* / /; /^$/d' > user/ps.sym
riscv64-linux-gnu-gcc -Wall -Werror -O -fno-omit-frame-pointer -ggdb -MD -mcmodel=medany -ffreestanding -fno-common -nostdlib -mno-relax -I. -fno-stack-protector -fno-pie -no-pie "-DPS"   -c -o user/priority.o user/priority.c
riscv64-linux-gnu-ld -z max-page-size=4096 -N -e main -Ttext 0 -o user/_priority user/priority.o user/ulib.o user/usys.o user/printf.o user/umalloc.o
riscv64-linux-gnu-objdump -S user/_priority > user/priority.asm
riscv64-linux-gnu-objdump -t user/_priority | sed '1,/SYMBOL TABLE/d; s/ .* / /; /^$/d' > user/priority.sym
riscv64-linux-gnu-gcc -Wall -Werror -O -fno-omit-frame-pointer -ggdb -MD -mcmodel=medany -ffreestanding -fno-common -nostdlib -mno-relax -I. -fno-stack-protector -fno-pie -no-pie "-DPS"   -c -o user/testps.o user/testps.c
riscv64-linux-gnu-ld -z max-page-size=4096 -N -e main -Ttext 0 -o user/_testps user/testps.o user/ulib.o user/usys.o user/printf.o user/umalloc.o
riscv64-linux-gnu-objdump -S user/_testps > user/testps.asm
riscv64-linux-gnu-objdump -t user/_testps | sed '1,/SYMBOL TABLE/d; s/ .* / /; /^$/d' > user/testps.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 "-DPS"   -c -o user/testfcfs.o user/testfcfs.c
riscv64-linux-gnu-ld -z max-page-size=4096 -N -e main -Ttext 0 -o user/_testfcfs user/testfcfs.o user/ulib.o user/usys.o user/printf.o user/umalloc.o
riscv64-linux-gnu-objdump -S user/_testfcfs > user/testfcfs.asm
riscv64-linux-gnu-objdump -t 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/_usertests user/_grind user/_wc user/_zombie user/_uniq user/_strace user/_time user/_ps user/_priority user/_testps user/_testfcfs
nmeta 46 (boot, super, log blocks 30 inode blocks 13, bitmap blocks 1) blocks 954 total 1000
balloc: first 783 blocks have been allocated
balloc: write bitmap block at sector 45
mkfs/mkfs fs.img uniqfile.txt 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
nmeta 46 (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
qemu-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-mmio-bus.0

xv6 kernel is booting

hart 2 starting
hart 1 starting
init: starting sh
$
```

Input files:



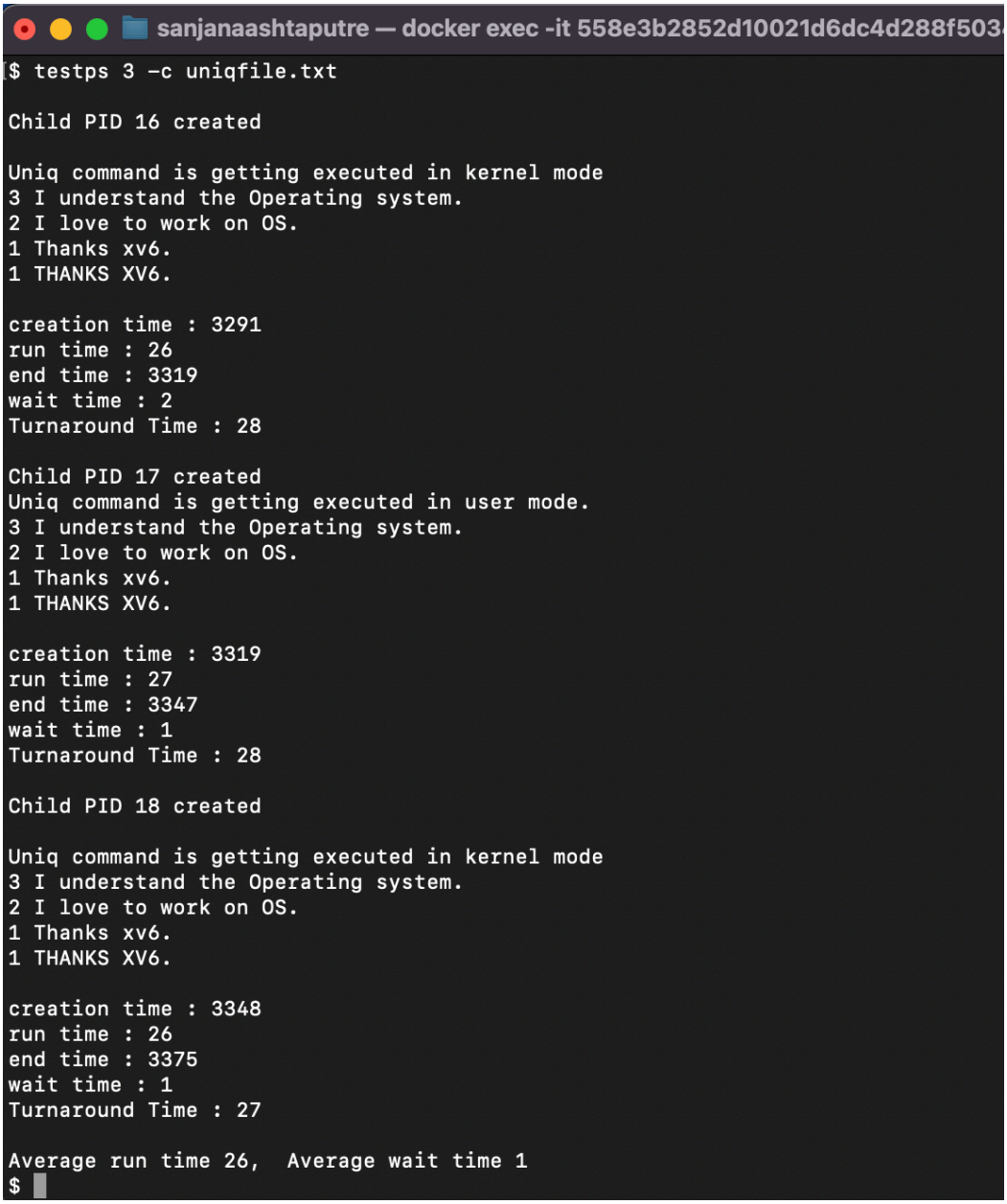
Output:

<program file> <number_of _processes> <Input_file>

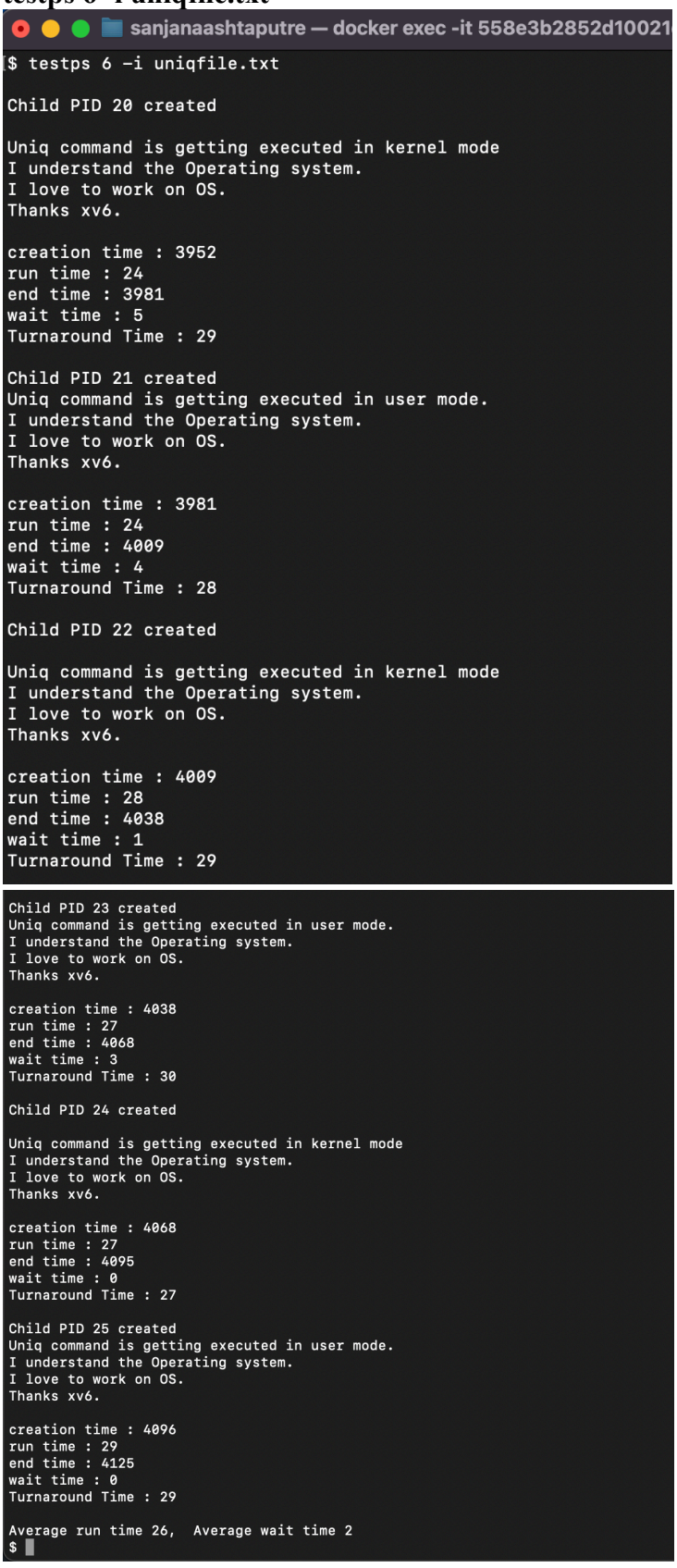
testps 4 uniqfile.txt

<program file> <number_of _processes> <-c/-d/-i> <Input_file>

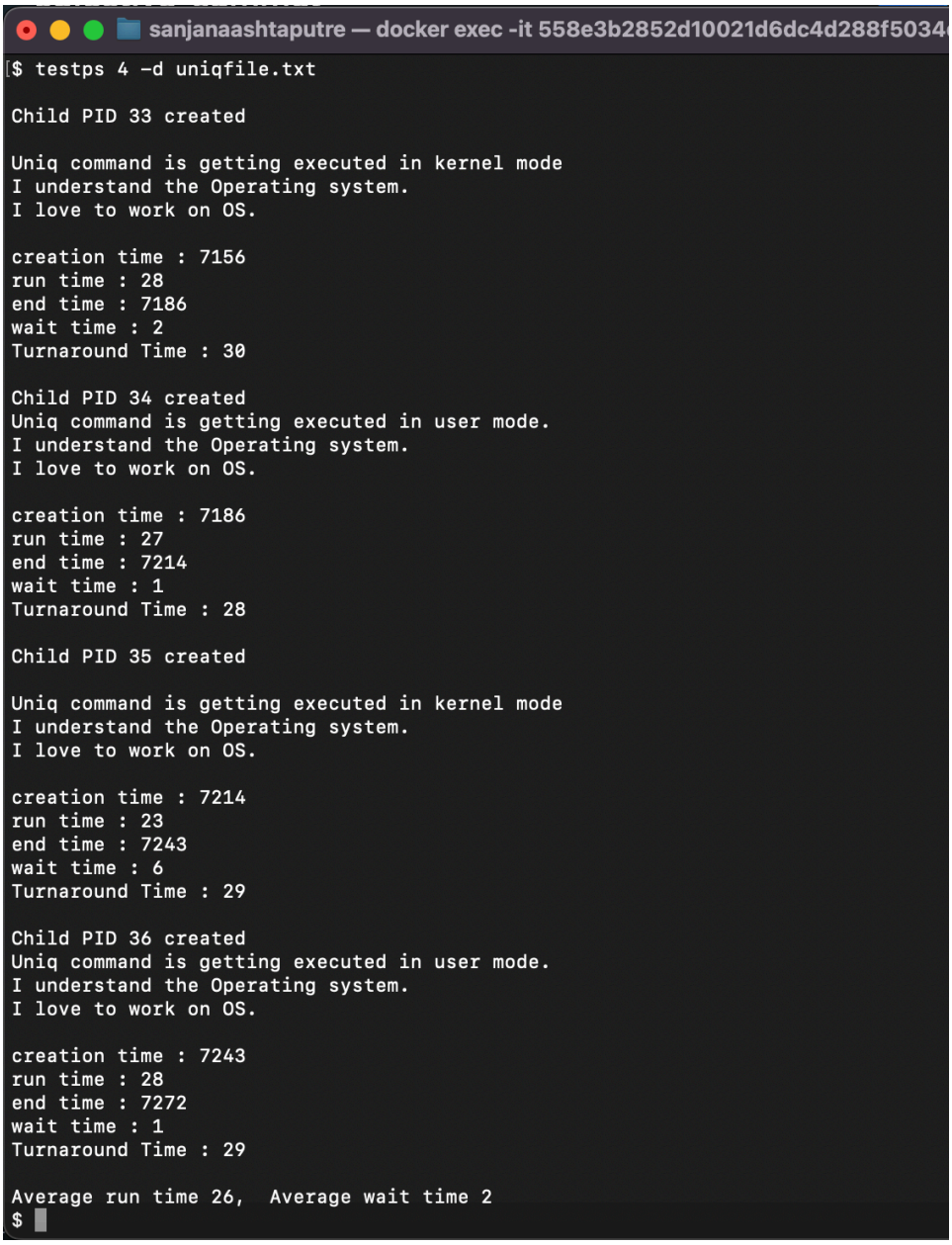
testps 3 -c uniqfile.txt



testps 6 -i uniqfile.txt



testps 4 -d uniqfile.txt



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

    char var_icd = 'o';

    if(argc < 2)
      n = 1;
    else
      n = atoi(argv[1]);
    if (n < 0 ||n > 20)
      n = 2;

    pid = 0;

    for ( k = 0; k < n; k++ ) {
      pid = fork ();
      if ( pid == 0 ) {
          printf("\nChild PID %d created\n",getpid()
          for(z = 0; z < 1000000000; z+=1){}

          for (int i = 2; i < argc; i++) {
              a=*argv[i];
```

```
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(fdk, buffer, sizeof(buffer))) > 0 )
    {
       uniq(n2, buffer, var_icd);
    }
    if (n2 < 0) {
         printf("Error: Unable to read file\n");
```

```
140                 }else{
141                     uniq_f(fdk,var_icd);
142                 }
143                 close(fdk);
144               }
145               f=0;
146             }
147         exit(0);
148         }else if ( pid < 0 ) {
149           printf("%d failed in fork!\n", getpid());
150         } else if (pid > 0) {
151           // parent
152           if(waitx(0,&rt,&wt) >= 0) {
153             trt += rt;
154             twt += wt;
155           }
156         }
157       }
158     printf("\nAverage run time %d,  Average wait time %d\n", trt / n, twt / n);
159     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  }
```

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.

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.

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  {
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");