README Project 3:

Name: Sudhamsh reddy dokuru
UID: U27437382
Course: Operating Systems
Faculty: John Templeton
There are two folders:

1. One contains fcfs implementation.
2. The other folder contains Priority order implementation.

list of file changed and changes made in FCFS and Priority(Common):
test.c(user): Made few changes to run multiple process at a time.
Usys.p(user):created a new user system call which sets priority to new processes created.
User.h(user): updated header file to accommodate fork77 user system call
Makefile: updated makefile(fork77 system call)
Sysfile.c: updated sysfile.c and added sys_fork77 system call
Syscall.h: updated and added SYS_fork77
Syscall.c: updated file to add fork77 system call
Proc.c: logic of fork77() and changed scheduler function
Defs.h:updated headed file for fork77 system call
Proc.h:added new attribute to struct proc to track priority.

Logic of FCFS:
Functions created in proc.c: DEFscheduler, fcfs, numberOfProcess

numberOfProcess :returns number of runnable processes
DEFscheduler: similar to round robin(similar previous xv6 scheduler)
Fcfs: runs process which has least creation time.
Note:when there are less than 3 runnable processes the scheduler runs DEFscheduler **else runs FCFS**

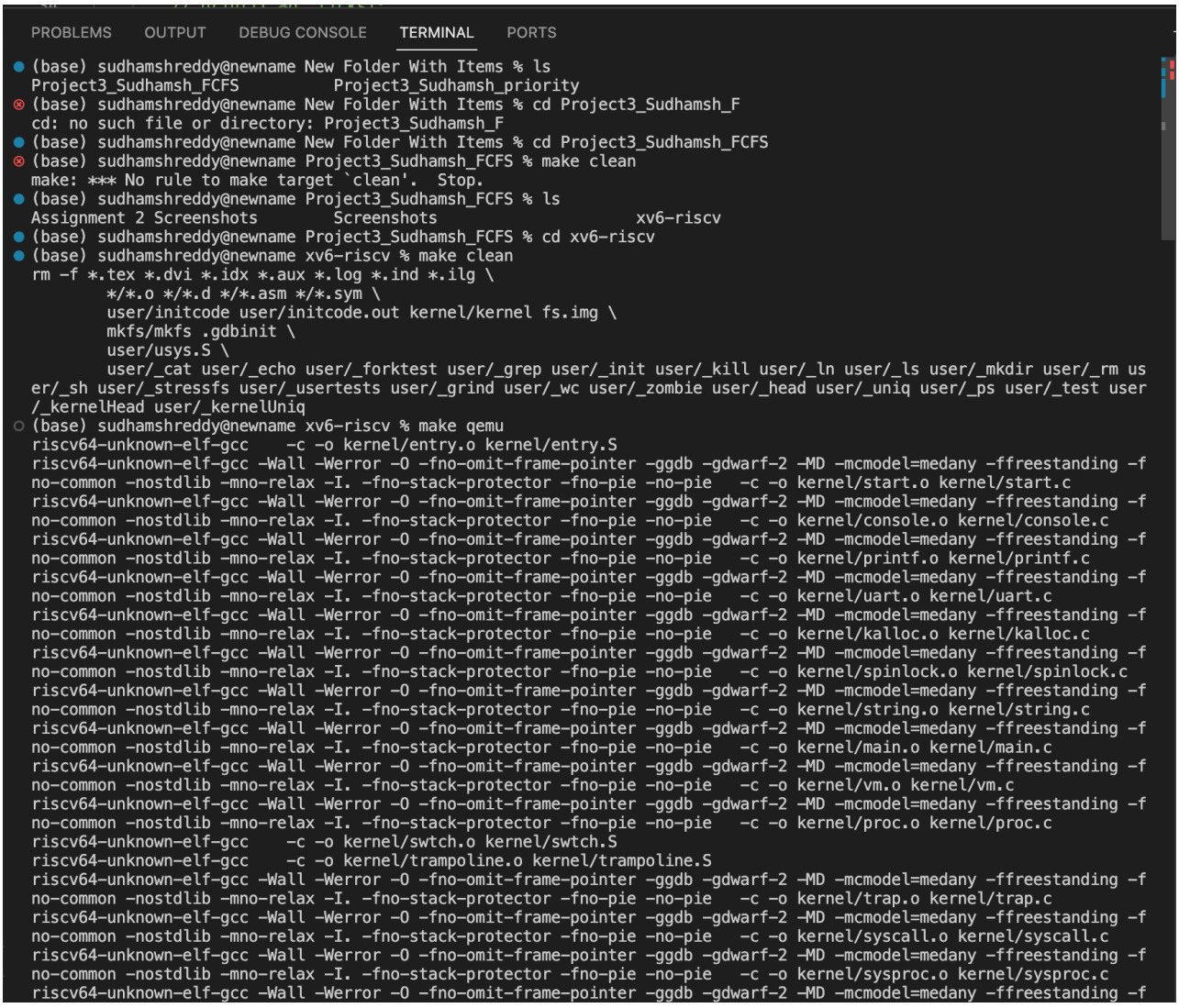
Logic of Priority:
Functions created in proc.c: DEFscheduler, fcfs, numberOfProcess

numberOfProcess :returns number of runnable processes
DEFscheduler: similar to round robin(similar previous xv6 scheduler)
Pscheduler: runs process which has highest priority time.
By default the priority is set 0;//least priority
Note:when there are less than 3 runnable processes the scheduler runs DEFscheduler **else runs** Pscheduler
To test Priority Scheduler:
Created a new system call named fork77 which sets the priority of a process when created. It is similar to fork system call the only difference is the assigning priority to process.

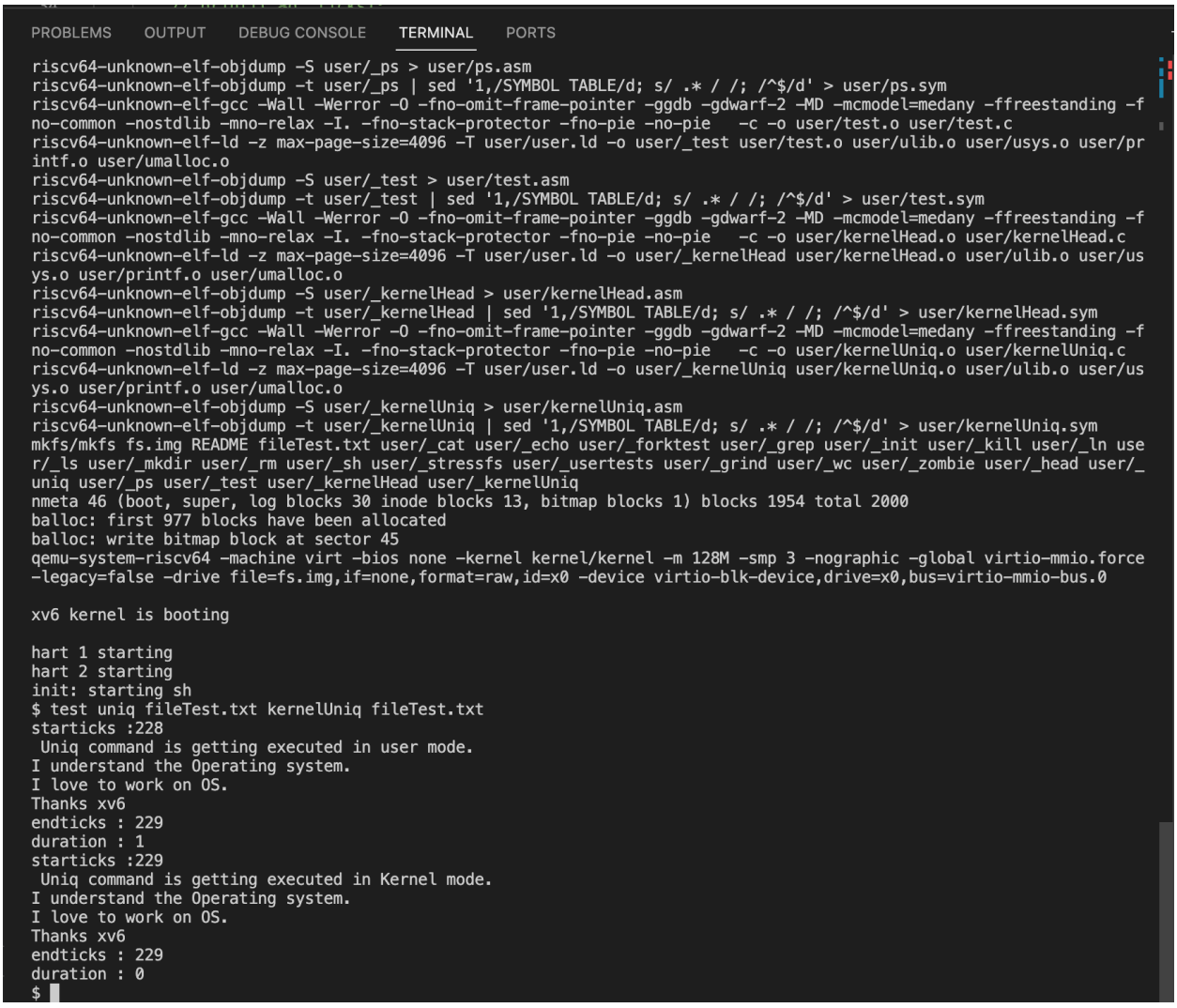
Wait calculation:
Made changes to ps system call from project 2 to print wait ticks
Wait ticks calculation: total run ticks – total ticks in running state
Total run ticks = end ticks – start ticks
Total ticks in running state = recorded ticks with the process entered running state and process exited the running

Test Case:

Make clean ,make qemu for FCFS:

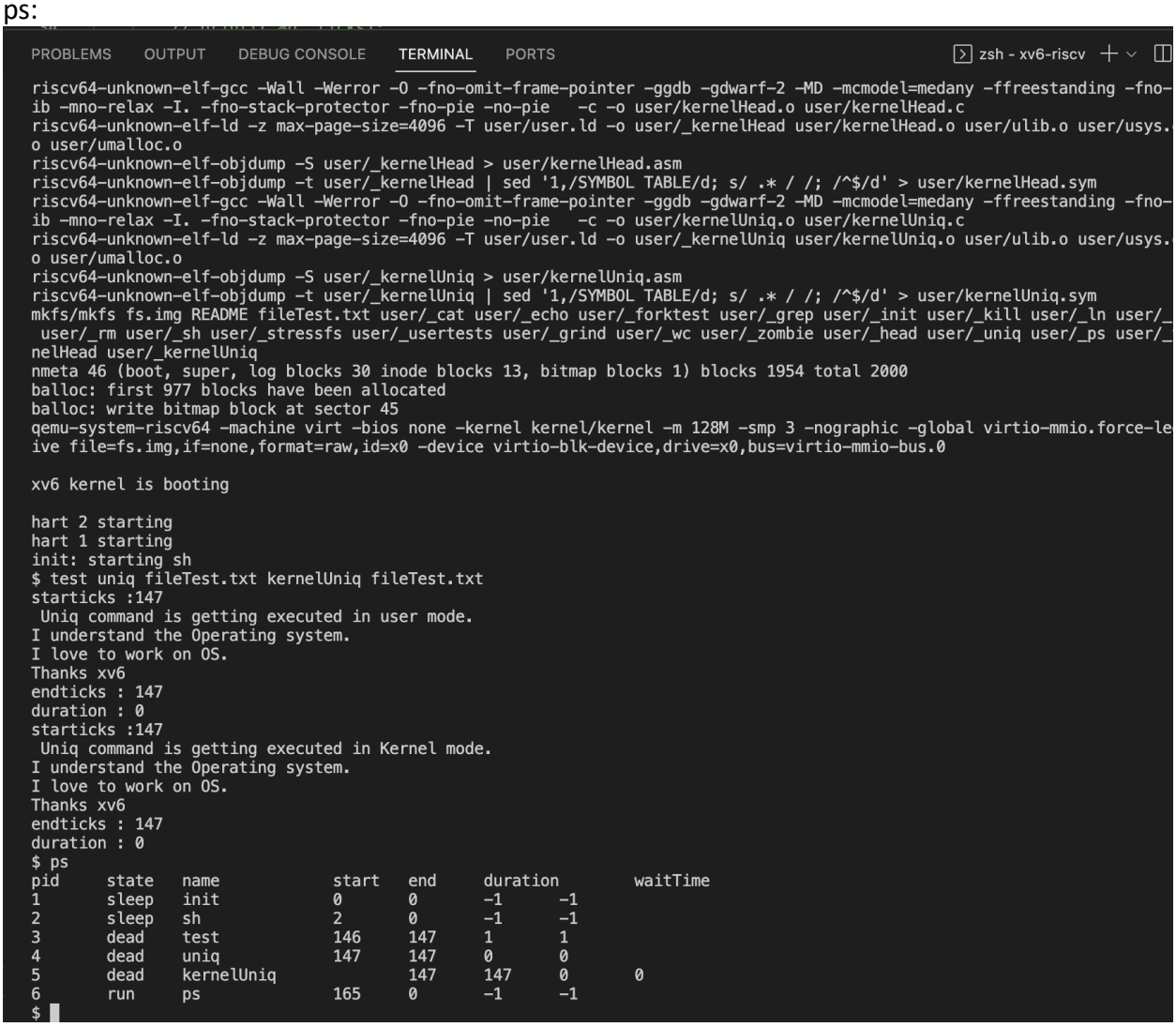


test uniq fileTest.txt kernelUniq fileTest.txt:



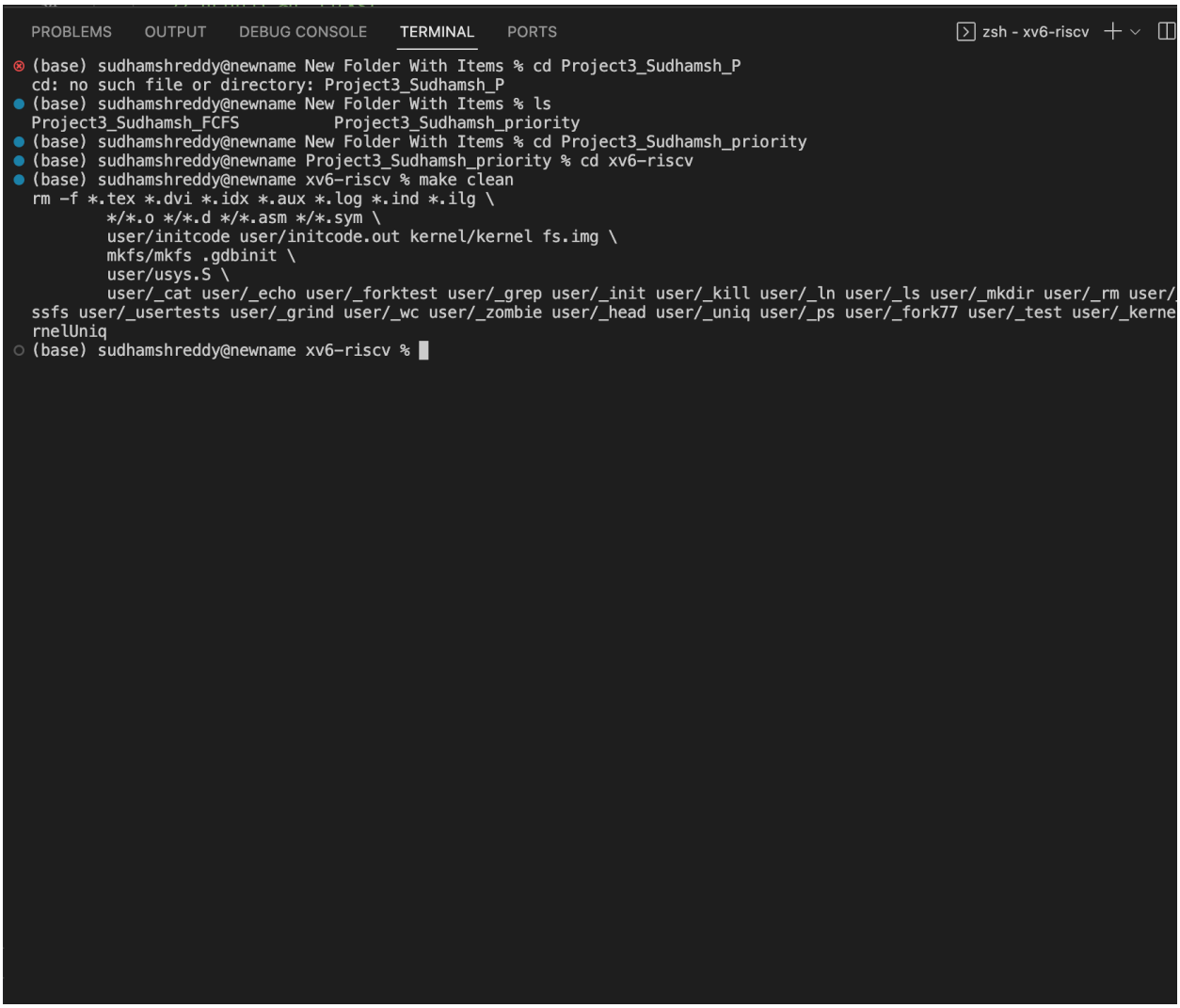
```
PROBLEMS  OUTPUT  DEBUG CONSOLE  TERMINAL  PORTS

riscv64-unknown-elf-objdump -S user/_ps > user/ps.asm
riscv64-unknown-elf-objdump -t user/_ps | sed '1,/SYMBOL TABLE/d; s/ .* / /; /^$/d' > user/ps.sym
riscv64-unknown-elf-gcc -Wall -Werror -O -fno-omit-frame-pointer -ggdb -gdwarf-2 -MD -mcmodel=medany -ffreestanding -f
no-common -nostdlib -mno-relax -I. -fno-stack-protector -fno-pie -no-pie   -c -o user/test.o user/test.c
riscv64-unknown-elf-ld -z max-page-size=4096 -T user/user.ld -o user/_test user/test.o user/ulib.o user/usys.o user/pr
intf.o user/umalloc.o
riscv64-unknown-elf-objdump -S user/_test > user/test.asm
riscv64-unknown-elf-objdump -t user/_test | sed '1,/SYMBOL TABLE/d; s/ .* / /; /^$/d' > user/test.sym
riscv64-unknown-elf-gcc -Wall -Werror -O -fno-omit-frame-pointer -ggdb -gdwarf-2 -MD -mcmodel=medany -ffreestanding -f
no-common -nostdlib -mno-relax -I. -fno-stack-protector -fno-pie -no-pie   -c -o user/kernelHead.o user/kernelHead.c
riscv64-unknown-elf-ld -z max-page-size=4096 -T user/user.ld -o user/_kernelHead user/kernelHead.o user/ulib.o user/us
ys.o user/printf.o user/umalloc.o
riscv64-unknown-elf-objdump -S user/_kernelHead > user/kernelHead.asm
riscv64-unknown-elf-objdump -t user/_kernelHead | sed '1,/SYMBOL TABLE/d; s/ .* / /; /^$/d' > user/kernelHead.sym
riscv64-unknown-elf-gcc -Wall -Werror -O -fno-omit-frame-pointer -ggdb -gdwarf-2 -MD -mcmodel=medany -ffreestanding -f
no-common -nostdlib -mno-relax -I. -fno-stack-protector -fno-pie -no-pie   -c -o user/kernelUniq.o user/kernelUniq.c
riscv64-unknown-elf-ld -z max-page-size=4096 -T user/user.ld -o user/_kernelUniq user/kernelUniq.o user/ulib.o user/us
ys.o user/printf.o user/umalloc.o
riscv64-unknown-elf-objdump -S user/_kernelUniq > user/kernelUniq.asm
riscv64-unknown-elf-objdump -t user/_kernelUniq | sed '1,/SYMBOL TABLE/d; s/ .* / /; /^$/d' > user/kernelUniq.sym
mkfs/mkfs fs.img README fileTest.txt user/_cat user/_echo user/_forktest user/_grep user/_init user/_kill user/_ln use
r/_ls user/_mkdir user/_rm user/_sh user/_stressfs user/_usertests user/_grind user/_wc user/_zombie user/_head user/_
uniq user/_ps user/_test user/_kernelHead user/_kernelUniq
nmeta 46 (boot, super, log blocks 30 inode blocks 13, bitmap blocks 1) blocks 1954 total 2000
balloc: first 977 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 -global virtio-mmio.force
-legacy=false -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 1 starting
hart 2 starting
init: starting sh
$ test uniq fileTest.txt kernelUniq fileTest.txt
starticks :228
 Uniq command is getting executed in user mode.
I understand the Operating system.
I love to work on OS.
Thanks xv6
endticks : 229
duration : 1
starticks :229
 Uniq command is getting executed in Kernel mode.
I understand the Operating system.
I love to work on OS.
Thanks xv6
endticks : 229
duration : 0
$
```

ps:

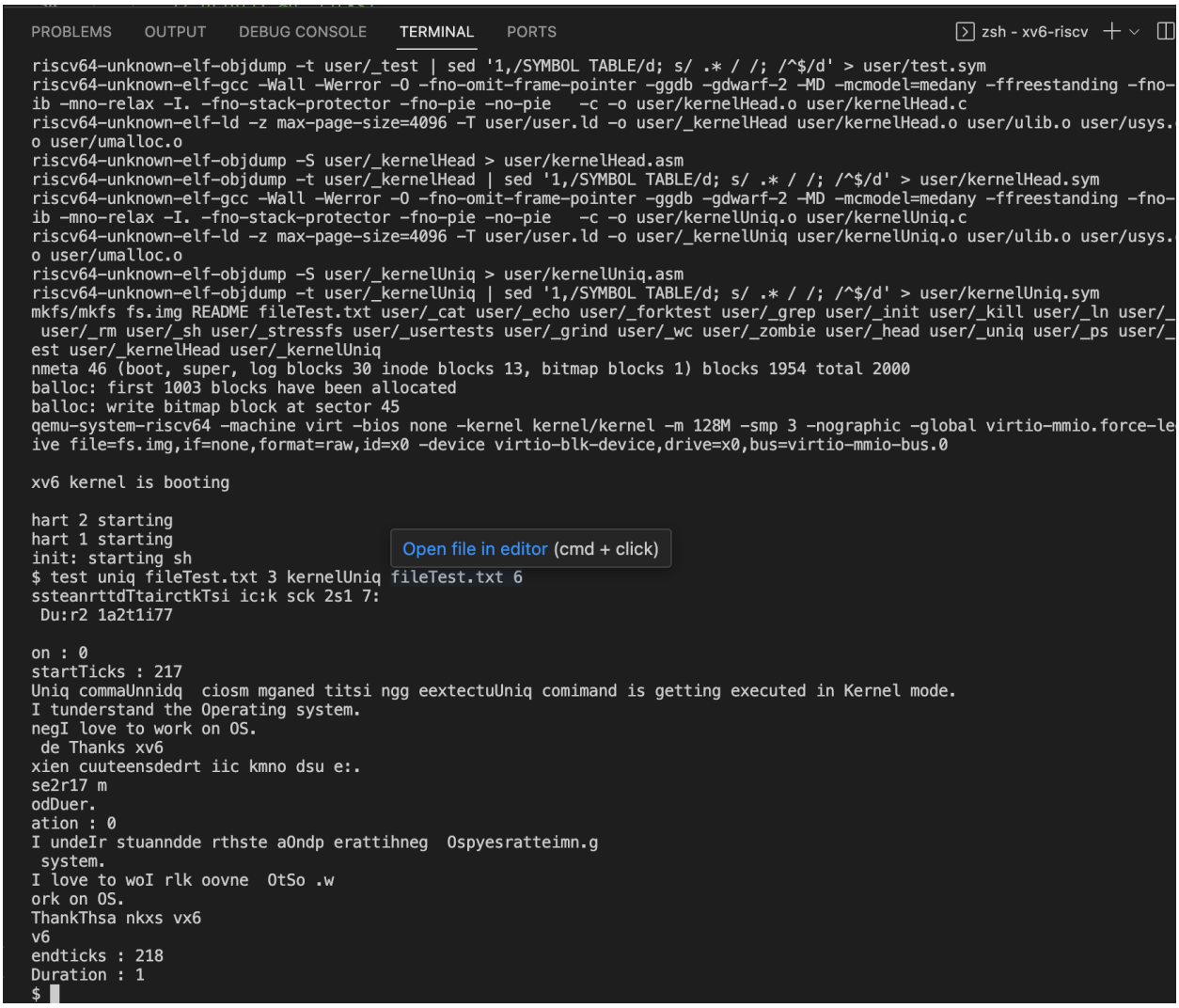


Test case for Priority:

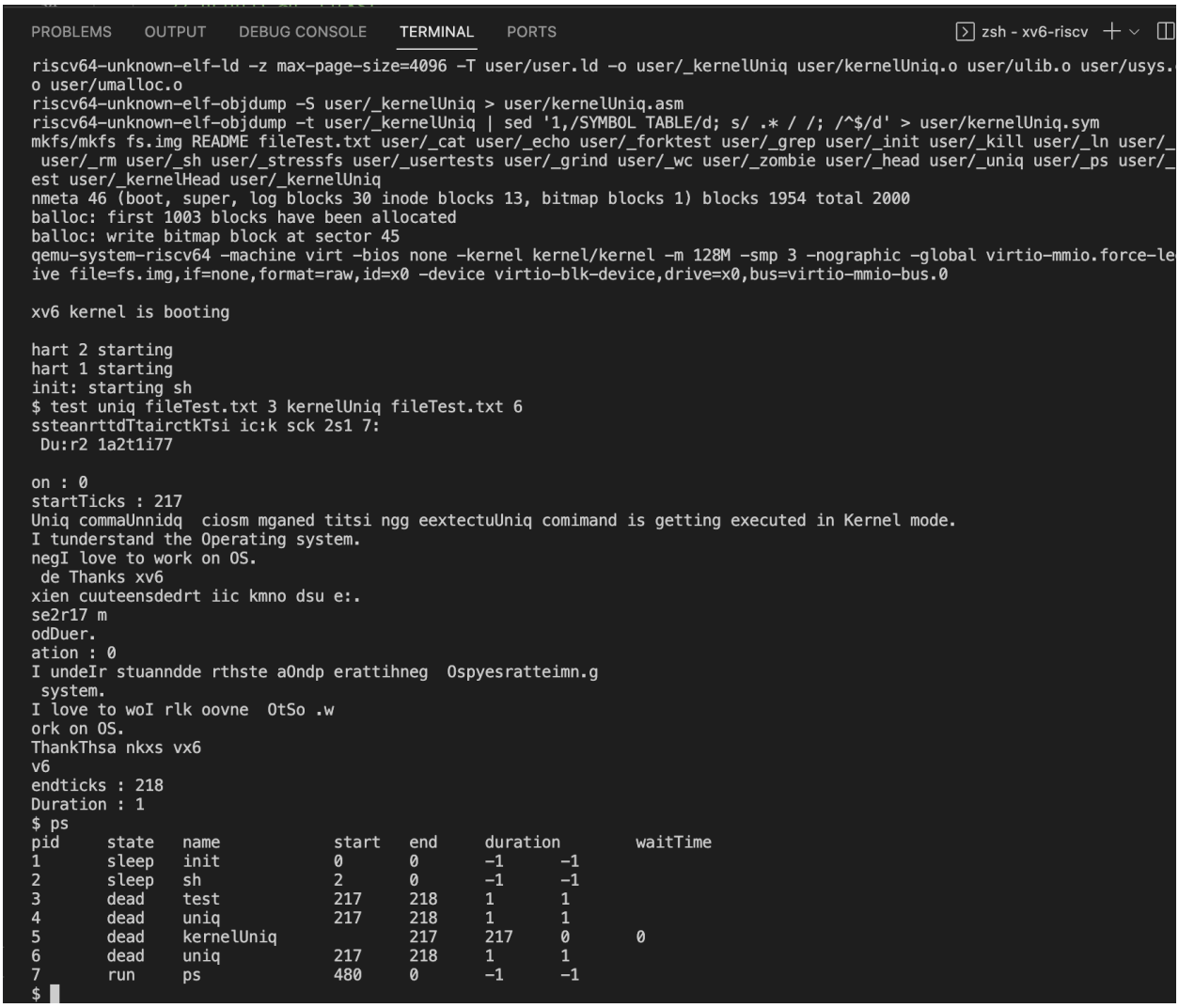
Make clean:



test uniq fileTest.txt 3 kernelUniq fileTest.txt 6:


PROBLEMS
OUTPUT
DEBUG CONSOLE
TERMINAL
PORTS
zsh - xv6-riscv
riscv64-unknown-elf-objdump -t user/_test | sed '1,/SYMBOL TABLE/d; s/ .* / /; /^$/d' > user/test.sym
riscv64-unknown-elf-gcc -Wall -Werror -O -fno-omit-frame-pointer -ggdb -gdwarf-2 -MD -mcmodel=medany -ffreestanding -fno-
ib -mno-relax -I. -fno-stack-protector -fno-pie -no-pie   -c -o user/kernelHead.o user/kernelHead.c
riscv64-unknown-elf-ld -z max-page-size=4096 -T user/user.ld -o user/_kernelHead user/kernelHead.o user/ulib.o user/usys.
o user/umalloc.o
riscv64-unknown-elf-objdump -S user/_kernelHead > user/kernelHead.asm
riscv64-unknown-elf-objdump -t user/_kernelHead | sed '1,/SYMBOL TABLE/d; s/ .* / /; /^$/d' > user/kernelHead.sym
riscv64-unknown-elf-gcc -Wall -Werror -O -fno-omit-frame-pointer -ggdb -gdwarf-2 -MD -mcmodel=medany -ffreestanding -fno-
ib -mno-relax -I. -fno-stack-protector -fno-pie -no-pie   -c -o user/kernelUniq.o user/kernelUniq.c
riscv64-unknown-elf-ld -z max-page-size=4096 -T user/user.ld -o user/_kernelUniq user/kernelUniq.o user/ulib.o user/usys.
o user/umalloc.o
riscv64-unknown-elf-objdump -S user/_kernelUniq > user/kernelUniq.asm
riscv64-unknown-elf-objdump -t user/_kernelUniq | sed '1,/SYMBOL TABLE/d; s/ .* / /; /^$/d' > user/kernelUniq.sym
mkfs/mkfs fs.img README fileTest.txt user/_cat user/_echo user/_forktest user/_grep user/_init user/_kill user/_ln user/_
 user/_rm user/_sh user/_stressfs user/_usertests user/_grind user/_wc user/_zombie user/_head user/_uniq user/_ps user/_
est user/_kernelHead user/_kernelUniq
nmeta 46 (boot, super, log blocks 30 inode blocks 13, bitmap blocks 1) blocks 1954 total 2000
balloc: first 1003 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 -global virtio-mmio.force-le
ive 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
Open file in editor (cmd + click)
$ test uniq fileTest.txt 3 kernelUniq fileTest.txt 6
ssteanrttdTtairctkTsi ic:k sck 2s1 7:
 Du:r2 1a2t1i77
on : 0
startTicks : 217
Uniq commaUnnidq ciosm mganed titsi ngg eextectuUniq comimand is getting executed in Kernel mode.
I tunderstand the Operating system.
negI love to work on OS.
 de Thanks xv6
xien cuuteensdedrt iic kmno dsu e:.
se2r17 m
odDuer.
ation : 0
I undeIr stuanndde rthste aOndp erattihneg Ospyesratteimn.g
 system.
I love to woI rlk oovne OtSo .w
ork on OS.
ThankThsa nkxs vx6
v6
endticks : 218
Duration : 1
$

Ps:



Note: **The XV6 in mac os supports multiprocessing due to which when we create multiple processes they run parallelly due to which the output is gibershed and in most of cases the wait time is near 0 ticks.**