Implementing ps on xv6

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

Code logic:

File: *ps.c*
The system call *ps()* is called in this file which is present in the user directory of xv6.

/xv6-riscv/user/ps.c

```c
#include "kernel/types.h"
#include "kernel/stat.h"
#include "user.h"


int main(int argc, char *argv[]) {
    ps();
        exit(0);
}

```

File: *defs.h*

```c
// proc.c
int             cpuid(void);
void            exit(int);
int             fork(void);
int             growproc(int);
void            proc_mapstacks(pagetable_t);
pagetable_t     proc_pagetable(struct proc *);
void            proc_freepagetable(pagetable_t, uint64);
int             kill(int);
int             killed(struct proc*);
void            setkilled(struct proc*);
struct cpu*     mycpu(void);
struct cpu*     getmycpu(void);
struct proc*    myproc();
void            procinit(void);
void            scheduler(void) __attribute__((noreturn));
void            sched(void);
void            sleep(void*, struct spinlock*);
void            userinit(void);
int             wait(uint64);
void            wakeup(void*);
void            yield(void);
int             either_copyout(int user_dst, uint64 dst, void *src, uint64 len);
int             either_copyin(void *dst, int user_src, uint64 src, uint64 len);
void            procdump(void);
int             waitext(uint64, uint64, uint64);
int             ps(void);
```

File: *proc.h*
Declared variables for process creation time (*ctime*), end time (etime), run time (*rtime*)and total time (*total_time*) in *proc* structure.

```c
// Per-process state
struct proc {
  struct spinlock lock;

  // p->lock must be held when using these:
  enum procstate state;        // Process state
  void *chan;                  // If non-zero, sleeping on chan
  int killed;                  // If non-zero, have been killed
  int xstate;                  // Exit status to be returned to parent's wait
  int pid;                     // Process ID

  // wait_lock must be held when using this:
  struct proc *parent;         // Parent process

  // these are private to the process, so p->lock need not be held.
  uint64 kstack;               // Virtual address of kernel stack
  uint64 sz;                   // Size of process memory (bytes)
  pagetable_t pagetable;       // User page table
  struct trapframe *trapframe; // data page for trampoline.S
  struct context context;      // swtch() here to run process
  struct file *ofile[NOFILE];  // Open files
  struct inode *cwd;           // Current directory
  char name[16];               // Process name (debugging)

  int ctime, etime, rtime;
  uint total_time;
};
```

File: *syscall.h*

```h
// System call numbers
#define SYS_fork    1
#define SYS_exit    2
#define SYS_wait    3
#define SYS_pipe    4
#define SYS_read    5
#define SYS_kill    6
#define SYS_exec    7
#define SYS_fstat   8
#define SYS_chdir   9
#define SYS_dup    10
#define SYS_getpid 11
#define SYS_sbrk   12
#define SYS_sleep  13
#define SYS_uptime 14
#define SYS_open   15
#define SYS_write  16
#define SYS_mknod  17
#define SYS_unlink 18
#define SYS_link   19
#define SYS_mkdir  20
#define SYS_close  21
#define SYS_uniq   22
#define SYS_head   23
#define SYS_waitext 24
#define SYS_ps 25
```

File: *syscall.c*

```c
extern uint64 sys_uniq(void);
extern uint64 sys_head(void);
extern uint64 sys_waitext(void);
extern uint64 sys_ps(void);

// An array mapping syscall numbers from syscall.h
// to the function that handles the system call.
static uint64 (*syscalls[])(void) = {
[SYS_fork]    sys_fork,
[SYS_exit]    sys_exit,
[SYS_wait]    sys_wait,
[SYS_pipe]    sys_pipe,
[SYS_read]    sys_read,
[SYS_kill]    sys_kill,
[SYS_exec]    sys_exec,
[SYS_fstat]   sys_fstat,
[SYS_chdir]   sys_chdir,
[SYS_dup]     sys_dup,
[SYS_getpid]  sys_getpid,
[SYS_sbrk]    sys_sbrk,
[SYS_sleep]   sys_sleep,
[SYS_uptime]  sys_uptime,
[SYS_open]    sys_open,
[SYS_write]   sys_write,
[SYS_mknod]   sys_mknod,
[SYS_unlink]  sys_unlink,
[SYS_link]    sys_link,
[SYS_mkdir]   sys_mkdir,
[SYS_close]   sys_close,
[SYS_uniq]    sys_uniq,
[SYS_head]    sys_head,
[SYS_waitext]   sys_waitext,
[SYS_ps]    sys_ps,
};
```

File: *usys.pl*

```
entry("kill");
entry("exec");
entry("open");
entry("mknod");
entry("unlink");
entry("fstat");
entry("link");
entry("mkdir");
entry("chdir");
entry("dup");
entry("getpid");
entry("sbrk");
entry("sleep");
entry("uptime");
entry("uniq");
entry("head");
entry("waitext");
entry("ps");
```

File: sysproc.c

```c
sys_waitext(void)
{
  uint64 rtime;
  uint64 p;

  argaddr(0, &wtime);

  argaddr(1, &rtime);

  argaddr(2, &p);

  return waitext(wtime, rtime, p);
}

int
sys_ps(void)
{
  return ps();
}
```

File: user.h

```h
int link(const char*, const char*);
int mkdir(const char*);
int chdir(const char*);
int dup(int);
int getpid(void);
char* sbrk(int);
int sleep(int);
int uptime(void);
int uniq(int n, char *buffer, char var_icd);
int head(int n, char *buffer, int lines);
int waitext(int *wtime, int *rtime, int *addr);
int ps(void);
```

File: *proc.c*

```c
allocproc(void)

  // time fields initializatio
  p->ctime = ticks;          //
  p->etime = 0;              //
  p->rtime = 0;              //
  p->total_time = 0;

  return p;
}
```

```c
scheduler(void)

    for(p = proc; p < &proc[NPROC]; p++) {
      acquire(&p->lock);
      if(p->state == RUNNABLE) {
        //update total time
        p->total_time += 1;
        // Switch to chosen process.  It is the
        // to release its lock and then reacquir
        // before jumping back to us.
        p->state = RUNNING;
        c->proc = p;
        swtch(&c->context, &p->context);
```

All the declared time fields are initialized in *allocproc()* function. The creation time is incremented with ticks, as the process starts. In *scheduler()* function, the *total_time* is incremented which stores the ticks whenever the process is running. In *ps()* function, the process name, pid, start time, process status, total time is displayed for unused (previous process that is now stored as unused), used, sleeping, running, runnable and zombie processes.

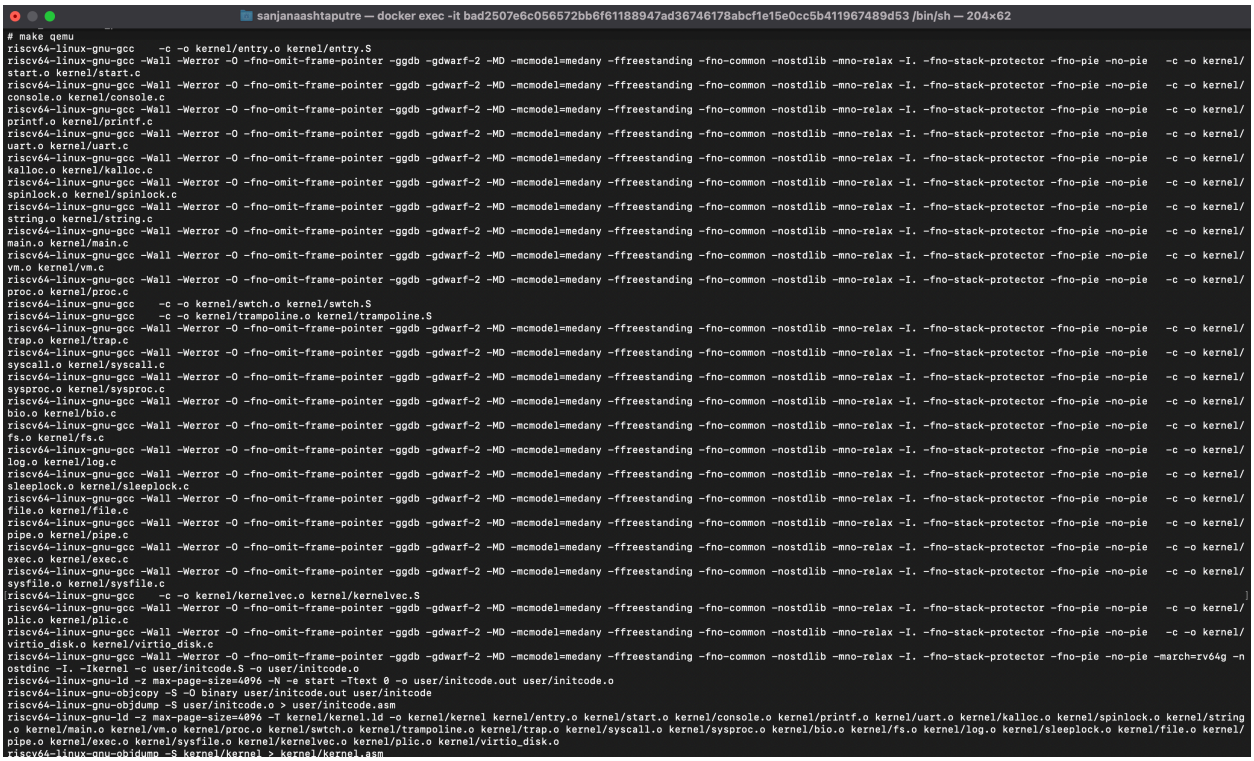
```c
int
ps()
{
struct proc *p;

//Loop over process table looking for process with pid.
acquire(&wait_lock);
printf("Name \t PID \t Status \t Start_time \t Total Scheduler time\n");
for(p = proc; p < &proc[NPROC]; p++){
  if(p->state == UNUSED && p->ctime != 0){
    printf("%s \t %d \t UNUSED ", p->name,p->pid);
    printf(" \t %d ",p->ctime);
    printf(" \t\t %d \n",p->total_time);
  }

    else if(p->state == USED){
    printf("%s \t %d \t USED ", p->name,p->pid);
    printf(" \t %d ",p->ctime);
    printf(" \t\t %d \n",p->total_time);
  }

  else if(p->state == SLEEPING){
    printf("%s \t %d \t SLEEPING ", p->name,p->pid);
    printf(" \t %d ",p->ctime);
    printf(" \t\t %d \n",p->total_time);
  }
    else if(p->state == RUNNING){
      printf("%s \t %d \t RUNNING ", p->name,p->pid);
      printf(" \t %d ",p->ctime);
      printf(" \t\t %d \n",p->total_time);
    }
    else if(p->state == RUNNABLE){
      printf("%s \t %d \t RUNNABLE ", p->name,p->pid);
      printf(" \t %d ",p->ctime);
      printf(" \t\t %d \n",p->total_time);
    }
    else if(p->state == ZOMBIE){
      printf("%s \t %d \t ZOMBIE ", p->name,p->pid);
      printf(" \t %d ",p->ctime);
      printf(" \t\t %d \n",p->total_time);
    }
}
release(&wait_lock);
return 25;
}
```

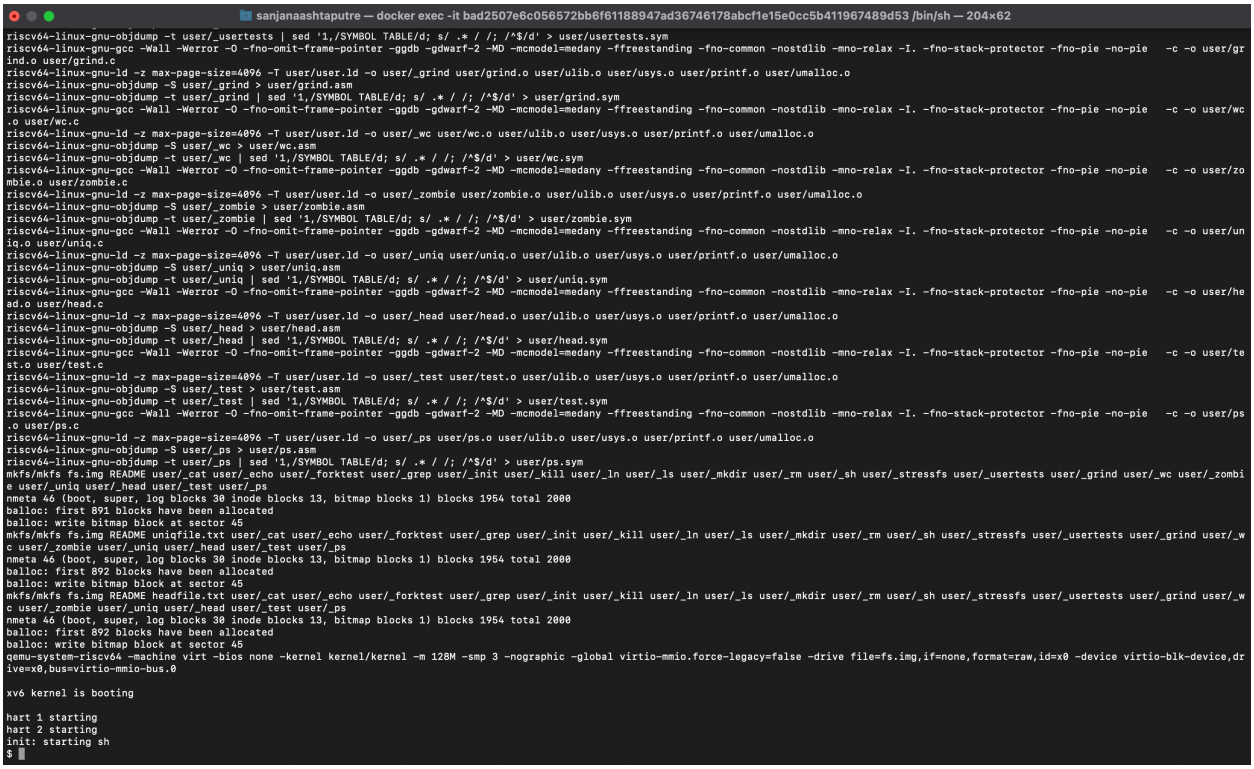
Steps to run the code and its output

make clean

```
sanjanaashtaputre — docker exec -it bad2507e6c056572bb6f61188947a...
# 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 us
er/_ls user/_mkdir user/_rm user/_sh user/_stressfs user/_usertests user/_grind
user/_wc user/_zombie user/_uniq user/_head user/_test user/_ps
#
```

make qemu


sanjanaashtaputre — docker exec -it bad2507e6c056572bb6f61188947ad36746178abcf1e15e0cc5b411967489d53 /bin/sh — 204×62
make qemu
riscv64-linux-gnu-gcc   -c -o kernel/entry.o kernel/entry.S
riscv64-linux-gnu-gcc -Wall -Werror -O -fno-omit-frame-pointer -ggdb -gdwarf-2 -MD -mcmodel=medany -ffreestanding -fno-common -nostdlib -mno-relax -I. -fno-stack-protector -fno-pie -no-pie   -c -o kernel/start.o kernel/start.c
riscv64-linux-gnu-gcc -Wall -Werror -O -fno-omit-frame-pointer -ggdb -gdwarf-2 -MD -mcmodel=medany -ffreestanding -fno-common -nostdlib -mno-relax -I. -fno-stack-protector -fno-pie -no-pie   -c -o kernel/console.o kernel/console.c
riscv64-linux-gnu-gcc -Wall -Werror -O -fno-omit-frame-pointer -ggdb -gdwarf-2 -MD -mcmodel=medany -ffreestanding -fno-common -nostdlib -mno-relax -I. -fno-stack-protector -fno-pie -no-pie   -c -o kernel/printf.o kernel/printf.c
riscv64-linux-gnu-gcc -Wall -Werror -O -fno-omit-frame-pointer -ggdb -gdwarf-2 -MD -mcmodel=medany -ffreestanding -fno-common -nostdlib -mno-relax -I. -fno-stack-protector -fno-pie -no-pie   -c -o kernel/uart.o kernel/uart.c
riscv64-linux-gnu-gcc -Wall -Werror -O -fno-omit-frame-pointer -ggdb -gdwarf-2 -MD -mcmodel=medany -ffreestanding -fno-common -nostdlib -mno-relax -I. -fno-stack-protector -fno-pie -no-pie   -c -o kernel/kalloc.o kernel/kalloc.c
riscv64-linux-gnu-gcc -Wall -Werror -O -fno-omit-frame-pointer -ggdb -gdwarf-2 -MD -mcmodel=medany -ffreestanding -fno-common -nostdlib -mno-relax -I. -fno-stack-protector -fno-pie -no-pie   -c -o kernel/spinlock.o kernel/spinlock.c
riscv64-linux-gnu-gcc -Wall -Werror -O -fno-omit-frame-pointer -ggdb -gdwarf-2 -MD -mcmodel=medany -ffreestanding -fno-common -nostdlib -mno-relax -I. -fno-stack-protector -fno-pie -no-pie   -c -o kernel/string.o kernel/string.c
riscv64-linux-gnu-gcc -Wall -Werror -O -fno-omit-frame-pointer -ggdb -gdwarf-2 -MD -mcmodel=medany -ffreestanding -fno-common -nostdlib -mno-relax -I. -fno-stack-protector -fno-pie -no-pie   -c -o kernel/main.o kernel/main.c
riscv64-linux-gnu-gcc -Wall -Werror -O -fno-omit-frame-pointer -ggdb -gdwarf-2 -MD -mcmodel=medany -ffreestanding -fno-common -nostdlib -mno-relax -I. -fno-stack-protector -fno-pie -no-pie   -c -o kernel/vm.o kernel/vm.c
riscv64-linux-gnu-gcc -Wall -Werror -O -fno-omit-frame-pointer -ggdb -gdwarf-2 -MD -mcmodel=medany -ffreestanding -fno-common -nostdlib -mno-relax -I. -fno-stack-protector -fno-pie -no-pie   -c -o kernel/proc.o kernel/proc.c
riscv64-linux-gnu-gcc   -c -o kernel/swtch.o kernel/swtch.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 -gdwarf-2 -MD -mcmodel=medany -ffreestanding -fno-common -nostdlib -mno-relax -I. -fno-stack-protector -fno-pie -no-pie   -c -o kernel/trap.o kernel/trap.c
riscv64-linux-gnu-gcc -Wall -Werror -O -fno-omit-frame-pointer -ggdb -gdwarf-2 -MD -mcmodel=medany -ffreestanding -fno-common -nostdlib -mno-relax -I. -fno-stack-protector -fno-pie -no-pie   -c -o kernel/syscall.o kernel/syscall.c
riscv64-linux-gnu-gcc -Wall -Werror -O -fno-omit-frame-pointer -ggdb -gdwarf-2 -MD -mcmodel=medany -ffreestanding -fno-common -nostdlib -mno-relax -I. -fno-stack-protector -fno-pie -no-pie   -c -o kernel/sysproc.o kernel/sysproc.c
riscv64-linux-gnu-gcc -Wall -Werror -O -fno-omit-frame-pointer -ggdb -gdwarf-2 -MD -mcmodel=medany -ffreestanding -fno-common -nostdlib -mno-relax -I. -fno-stack-protector -fno-pie -no-pie   -c -o kernel/bio.o kernel/bio.c
riscv64-linux-gnu-gcc -Wall -Werror -O -fno-omit-frame-pointer -ggdb -gdwarf-2 -MD -mcmodel=medany -ffreestanding -fno-common -nostdlib -mno-relax -I. -fno-stack-protector -fno-pie -no-pie   -c -o kernel/fs.o kernel/fs.c
riscv64-linux-gnu-gcc -Wall -Werror -O -fno-omit-frame-pointer -ggdb -gdwarf-2 -MD -mcmodel=medany -ffreestanding -fno-common -nostdlib -mno-relax -I. -fno-stack-protector -fno-pie -no-pie   -c -o kernel/log.o kernel/log.c
riscv64-linux-gnu-gcc -Wall -Werror -O -fno-omit-frame-pointer -ggdb -gdwarf-2 -MD -mcmodel=medany -ffreestanding -fno-common -nostdlib -mno-relax -I. -fno-stack-protector -fno-pie -no-pie   -c -o kernel/sleeplock.o kernel/sleeplock.c
riscv64-linux-gnu-gcc -Wall -Werror -O -fno-omit-frame-pointer -ggdb -gdwarf-2 -MD -mcmodel=medany -ffreestanding -fno-common -nostdlib -mno-relax -I. -fno-stack-protector -fno-pie -no-pie   -c -o kernel/file.o kernel/file.c
riscv64-linux-gnu-gcc -Wall -Werror -O -fno-omit-frame-pointer -ggdb -gdwarf-2 -MD -mcmodel=medany -ffreestanding -fno-common -nostdlib -mno-relax -I. -fno-stack-protector -fno-pie -no-pie   -c -o kernel/pipe.o kernel/pipe.c
riscv64-linux-gnu-gcc -Wall -Werror -O -fno-omit-frame-pointer -ggdb -gdwarf-2 -MD -mcmodel=medany -ffreestanding -fno-common -nostdlib -mno-relax -I. -fno-stack-protector -fno-pie -no-pie   -c -o kernel/exec.o kernel/exec.c
riscv64-linux-gnu-gcc -Wall -Werror -O -fno-omit-frame-pointer -ggdb -gdwarf-2 -MD -mcmodel=medany -ffreestanding -fno-common -nostdlib -mno-relax -I. -fno-stack-protector -fno-pie -no-pie   -c -o kernel/sysfile.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 -gdwarf-2 -MD -mcmodel=medany -ffreestanding -fno-common -nostdlib -mno-relax -I. -fno-stack-protector -fno-pie -no-pie   -c -o kernel/plic.o kernel/plic.c
riscv64-linux-gnu-gcc -Wall -Werror -O -fno-omit-frame-pointer -ggdb -gdwarf-2 -MD -mcmodel=medany -ffreestanding -fno-common -nostdlib -mno-relax -I. -fno-stack-protector -fno-pie -no-pie   -c -o kernel/virtio_disk.o kernel/virtio_disk.c
riscv64-linux-gnu-gcc -Wall -Werror -O -fno-omit-frame-pointer -ggdb -gdwarf-2 -MD -mcmodel=medany -ffreestanding -fno-common -nostdlib -mno-relax -I. -fno-stack-protector -fno-pie -no-pie -march=rv64g -nostdinc -I. -Ikernel -c user/initcode.S -o user/initcode.o
riscv64-linux-gnu-ld -z max-page-size=4096 -N -e start -Ttext 0 -o user/initcode.out user/initcode.o
riscv64-linux-gnu-objcopy -S -O binary user/initcode.out user/initcode
riscv64-linux-gnu-objdump -S user/initcode.o > user/initcode.asm
riscv64-linux-gnu-ld -z max-page-size=4096 -T kernel/kernel.ld -o kernel/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/swtch.o kernel/trampoline.o kernel/trap.o kernel/syscall.o kernel/sysproc.o kernel/bio.o kernel/fs.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

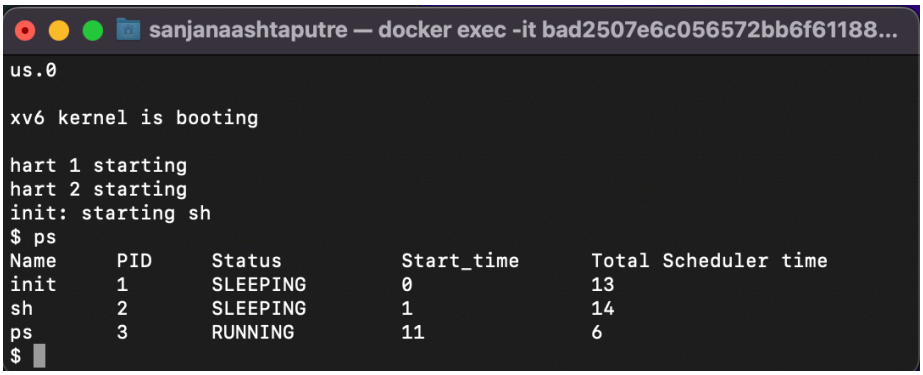

sanjanaashtaputre — docker exec -it bad2507e6c056572bb6f61188947ad36746178abcf1e15e0cc5b411967489d53 /bin/sh — 204×62
riscv64-linux-gnu-objdump -t user/_usertests | sed '1,/SYMBOL TABLE/d; s/ .* / /; /^$/d' > user/usertests.sym
riscv64-linux-gnu-gcc -Wall -Werror -O -fno-omit-frame-pointer -ggdb -gdwarf-2 -MD -mcmodel=medany -ffreestanding -fno-common -nostdlib -mno-relax -I. -fno-stack-protector -fno-pie -no-pie   -c -o user/grind.o user/grind.c
riscv64-linux-gnu-ld -z max-page-size=4096 -T user/user.ld -o user/_grind user/grind.o user/ulib.o user/usys.o user/printf.o user/umalloc.o
riscv64-linux-gnu-objdump -S user/_grind > user/grind.asm
riscv64-linux-gnu-objdump -t user/_grind | sed '1,/SYMBOL TABLE/d; s/ .* / /; /^$/d' > user/grind.sym
riscv64-linux-gnu-gcc -Wall -Werror -O -fno-omit-frame-pointer -ggdb -gdwarf-2 -MD -mcmodel=medany -ffreestanding -fno-common -nostdlib -mno-relax -I. -fno-stack-protector -fno-pie -no-pie   -c -o user/wc.o user/wc.c
riscv64-linux-gnu-ld -z max-page-size=4096 -T user/user.ld -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/ .* / /; /^$/d' > user/wc.sym
riscv64-linux-gnu-gcc -Wall -Werror -O -fno-omit-frame-pointer -ggdb -gdwarf-2 -MD -mcmodel=medany -ffreestanding -fno-common -nostdlib -mno-relax -I. -fno-stack-protector -fno-pie -no-pie   -c -o user/zombie.o user/zombie.c
riscv64-linux-gnu-ld -z max-page-size=4096 -T user/user.ld -o user/_zombie user/zombie.o user/ulib.o user/usys.o user/printf.o user/umalloc.o
riscv64-linux-gnu-objdump -S user/_zombie > user/zombie.asm
riscv64-linux-gnu-objdump -t user/_zombie | sed '1,/SYMBOL TABLE/d; s/ .* / /; /^$/d' > user/zombie.sym
riscv64-linux-gnu-gcc -Wall -Werror -O -fno-omit-frame-pointer -ggdb -gdwarf-2 -MD -mcmodel=medany -ffreestanding -fno-common -nostdlib -mno-relax -I. -fno-stack-protector -fno-pie -no-pie   -c -o user/uniq.o user/uniq.c
riscv64-linux-gnu-ld -z max-page-size=4096 -T user/user.ld -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/ .* / /; /^$/d' > user/uniq.sym
riscv64-linux-gnu-gcc -Wall -Werror -O -fno-omit-frame-pointer -ggdb -gdwarf-2 -MD -mcmodel=medany -ffreestanding -fno-common -nostdlib -mno-relax -I. -fno-stack-protector -fno-pie -no-pie   -c -o user/head.o user/head.c
riscv64-linux-gnu-ld -z max-page-size=4096 -T user/user.ld -o user/_head user/head.o user/ulib.o user/usys.o user/printf.o user/umalloc.o
riscv64-linux-gnu-objdump -S user/_head > user/head.asm
riscv64-linux-gnu-objdump -t user/_head | sed '1,/SYMBOL TABLE/d; s/ .* / /; /^$/d' > user/head.sym
riscv64-linux-gnu-gcc -Wall -Werror -O -fno-omit-frame-pointer -ggdb -gdwarf-2 -MD -mcmodel=medany -ffreestanding -fno-common -nostdlib -mno-relax -I. -fno-stack-protector -fno-pie -no-pie   -c -o user/test.o user/test.c
riscv64-linux-gnu-ld -z max-page-size=4096 -T user/user.ld -o user/_test user/test.o user/ulib.o user/usys.o user/printf.o user/umalloc.o
riscv64-linux-gnu-objdump -S user/_test > user/test.asm
riscv64-linux-gnu-objdump -t user/_test | sed '1,/SYMBOL TABLE/d; s/ .* / /; /^$/d' > user/test.sym
riscv64-linux-gnu-gcc -Wall -Werror -O -fno-omit-frame-pointer -ggdb -gdwarf-2 -MD -mcmodel=medany -ffreestanding -fno-common -nostdlib -mno-relax -I. -fno-stack-protector -fno-pie -no-pie   -c -o user/ps.o user/ps.c
riscv64-linux-gnu-ld -z max-page-size=4096 -T user/user.ld -o user/_ps user/ps.o user/ulib.o user/usys.o user/printf.o user/umalloc.o
riscv64-linux-gnu-objdump -S user/_ps > user/ps.asm
riscv64-linux-gnu-objdump -t user/_ps | sed '1,/SYMBOL TABLE/d; s/ .* / /; /^$/d' > user/ps.sym
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/_uniq user/_head user/_test user/_ps
nmeta 46 (boot, super, log blocks 30 inode blocks 13, bitmap blocks 1) blocks 1954 total 2000
balloc: first 891 blocks have been allocated
balloc: write bitmap block at sector 45
mkfs/mkfs fs.img README 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/_head user/_test user/_ps
nmeta 46 (boot, super, log blocks 30 inode blocks 13, bitmap blocks 1) blocks 1954 total 2000
balloc: first 892 blocks have been allocated
balloc: write bitmap block at sector 45
mkfs/mkfs fs.img README headfile.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/_head user/_test user/_ps
nmeta 46 (boot, super, log blocks 30 inode blocks 13, bitmap blocks 1) blocks 1954 total 2000
balloc: first 892 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

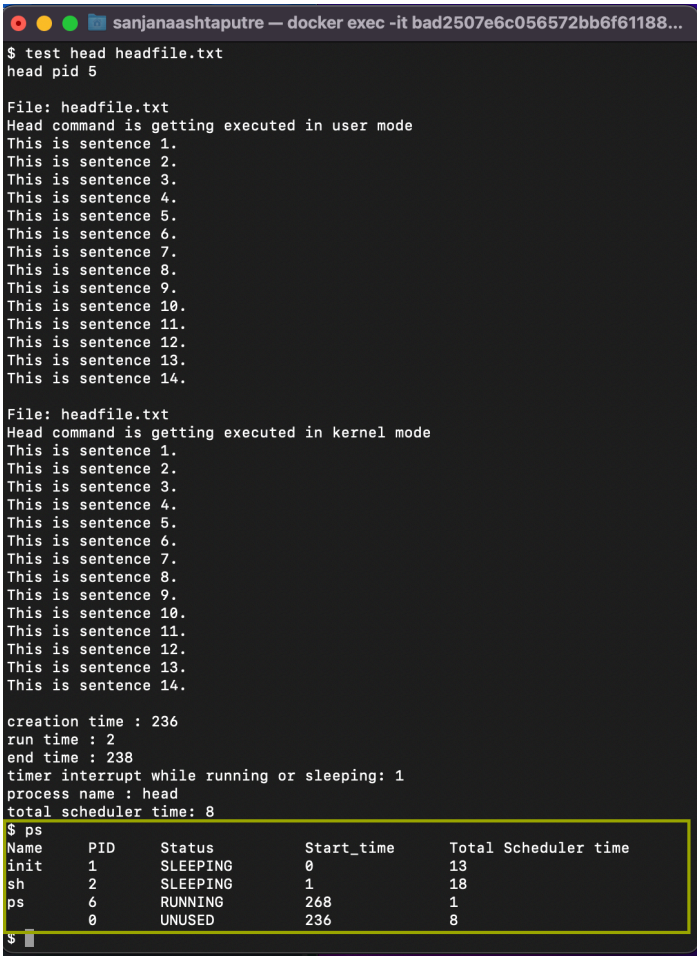
hart 1 starting
hart 2 starting
init: starting sh
$

Output:

ps as the first command with no previous user command.



ps after a command is executed. On successful execution, the process is stored with unused status.



When *ps()* is added in the test.c file.

/xv6-riscv/user/test.c

```
11      for(int i=1; i<argc; i++){
12        argv2[i-1] = argv[i];
13      }
14      ps();
15
16      pid = fork();
17      if (pid == 0) {
18        // Child process
19        exec(argv[1], argv2);
20        printf("Failed to execute %s\n", argv[1]);
21        exit(1);
22      } else if (pid > 0) {
23        printf("%s pid %d \n", argv[1], pid);
24        // Parent process
25         waitext(&wt, &rt, &st);
26      } else {
27        printf("Failed to fork %s\n", argv[1]);
28      }
29    ps();
30
31    exit(0);
32  }
33
```

sanjanaashtaputre — docker exec -it bad2507e6c056572bb6f61188...

```
$ test head -5 headfile.txt
Name     PID     Status          Start_time      Total Scheduler time
init     1       SLEEPING        0               25
sh       2       SLEEPING        2               13
test     3       RUNNING         77              6
head pid 4

File: headfile.txt
Head command is getting executed in user mode
This is sentence 1.
This is sentence 2.
This is sentence 3.
This is sentence 4.
This is sentence 5.

File: headfile.txt
Head command is getting executed in kernel mode
This is sentence 1.
This is sentence 2.
This is sentence 3.
This is sentence 4.
This is sentence 5.

creation time : 78
run time : 1
end time : 79
timer interrupt while running or sleeping: 1
process name : head
total scheduler time: 8
Name     PID     Status          Start_time      Total Scheduler time
init     1       SLEEPING        0               25
sh       2       SLEEPING        2               13
test     3       RUNNING         77              7
         0       UNUSED          78              8
$
```