LAPORAN TUGAS AKHIR

Mata Kuliah: Sistem Operasi
Semester: Genap / Tahun Ajaran 2024–2025
Nama: Nur Dini Handayani
NIM: 240202876
Modul yang Dikerjakan: Modul 2 – Penjadwalan CPU Lanjutan (Priority Scheduling Non-Preemptive)

■ Modul 2 — Penjadwalan CPU Lanjutan (Priority Scheduling Non-Preemptive)

Tujuan: Mengubah algoritma penjadwalan proses di xv6-public dari Round Robin menjadi Non-Preemptive Priority Scheduling.

■■ Ringkasan Implementasi:

- Menambahkan field `priority` pada setiap proses di `proc.h`
- Membuat syscall baru `set_priority(int)`
- Memodifikasi scheduler di `proc.c` untuk memilih proses dengan prioritas tertinggi (angka lebih kecil)
- Membuat program uji `ptest.c` untuk memverifikasi bahwa proses dengan prioritas lebih tinggi dijalankan terlebih dahulu

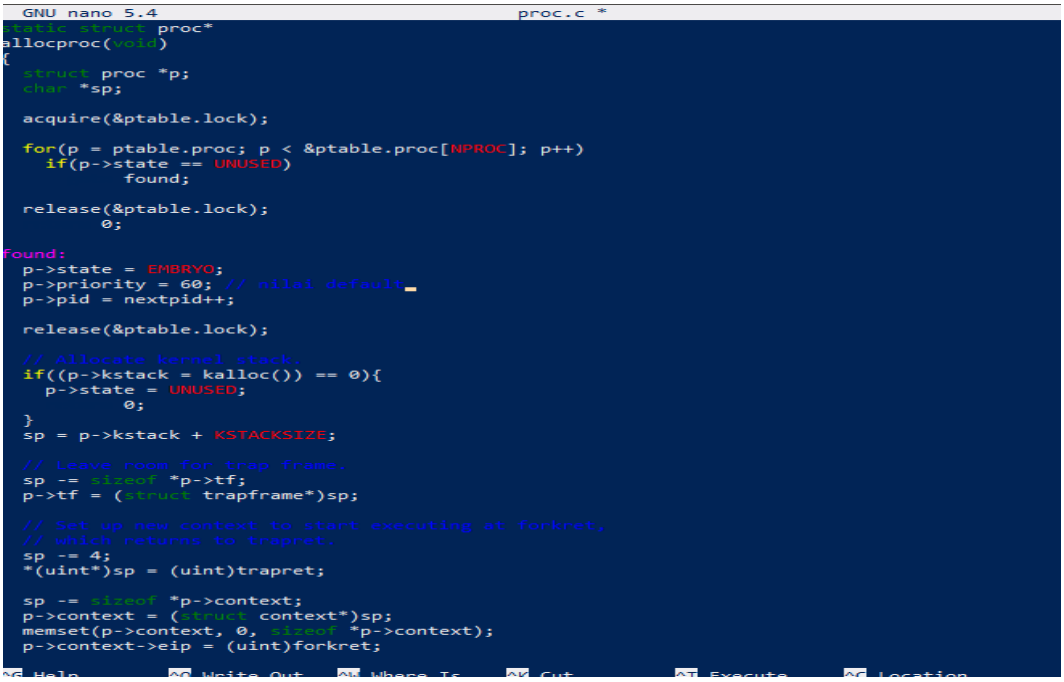
■ Validasi:

Program uji `ptest` menunjukkan proses dengan prioritas tertinggi (angka terkecil) dijalankan lebih dahulu secara non-preemptive.

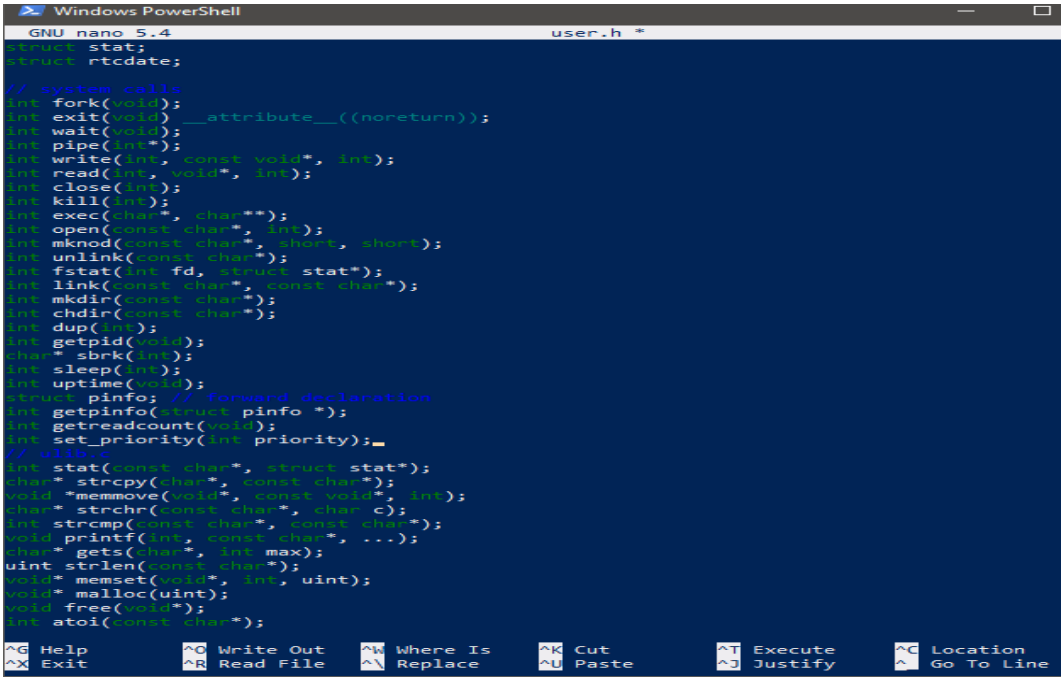
■ Kesimpulan:

Sistem berhasil diubah menjadi menggunakan Non-Preemptive Priority Scheduling. Proses dengan prioritas lebih tinggi akan didahulukan meski tidak terjadi preemption.

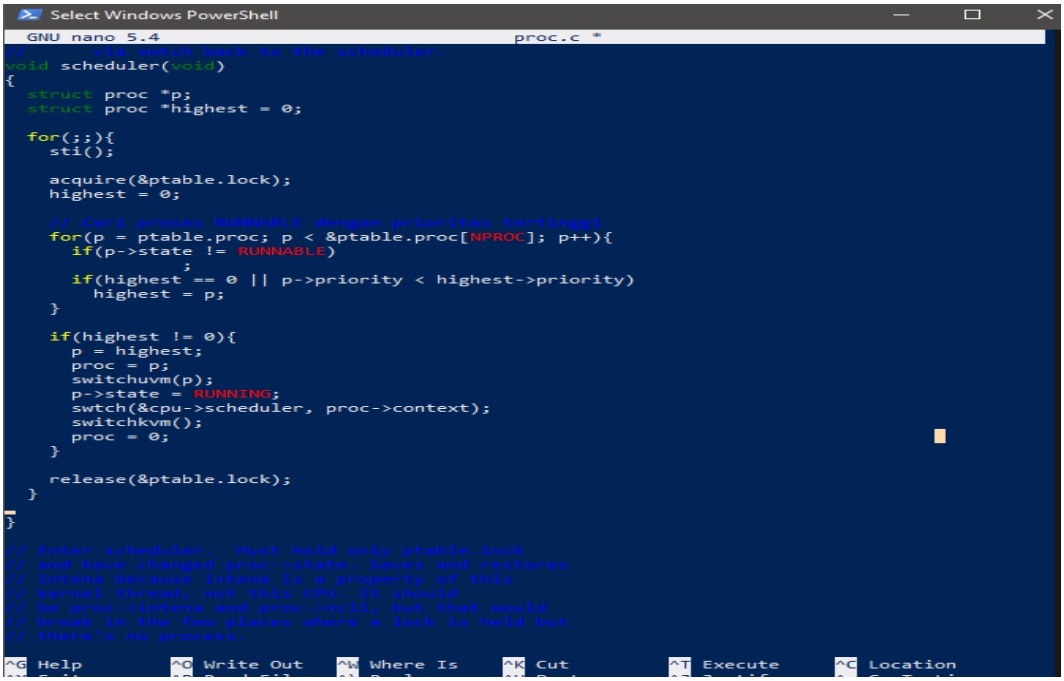
■ Dokumentasi Hasil Uji:



```
 GNU nano 5.4                                  proc.c *
static struct proc*
allocproc(void)
{
  struct proc *p;
  char *sp;

  acquire(&ptable.lock);

  for(p = ptable.proc; p < &ptable.proc[NPROC]; p++)
    if(p->state == UNUSED)
            found;

  release(&ptable.lock);
          0;

found:
  p->state = EMBRYO;
  p->priority = 60; // nilai default
  p->pid = nextpid++;

  release(&ptable.lock);

  // Allocate kernel stack.
  if((p->kstack = kalloc()) == 0){
    p->state = UNUSED;
            0;
  }
  sp = p->kstack + KSTACKSIZE;

  // Leave room for trap frame.
  sp -= sizeof *p->tf;
  p->tf = (struct trapframe*)sp;

  // Set up new context to start executing at forkret,
  // which returns to trapret.
  sp -= 4;
  *(uint*)sp = (uint)trapret;

  sp -= sizeof *p->context;
  p->context = (struct context*)sp;
  memset(p->context, 0, sizeof *p->context);
  p->context->eip = (uint)forkret;
```



```
Windows PowerShell
 GNU nano 5.4                                  user.h *
struct stat;
struct rtcdate;

// system calls
int fork(void);
int exit(void) __attribute__((noreturn));
int wait(void);
int pipe(int*);
int write(int, const void*, int);
int read(int, void*, int);
int close(int);
int kill(int);
int exec(char*, char**);
int open(const char*, int);
int mknod(const char*, short, short);
int unlink(const char*);
int fstat(int fd, struct stat*);
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);
struct pinfo; // forward declaration
int getpinfo(struct pinfo *);
int getreadcount(void);
int set_priority(int priority);
// ulib.c
int stat(const char*, struct stat*);
char* strcpy(char*, const char*);
void *memmove(void*, const void*, int);
char* strchr(const char*, char c);
int strcmp(const char*, const char*);
void printf(int, const char*, ...);
char* gets(char*, int max);
uint strlen(const char*);
void* memset(void*, int, uint);
void* malloc(uint);
void free(void*);
int atoi(const char*);

^G Help        ^O Write Out   ^W Where Is    ^K Cut         ^T Execute     ^C Location
^X Exit        ^R Read File   ^\ Replace     ^U Paste       ^J Justify     ^_ Go To Line
```


Select Windows PowerShell
GNU nano 5.4 proc.c *
// via swtch back to the scheduler.
void scheduler(void)
{
struct proc *p;
struct proc *highest = 0;
for(;;){
sti();
acquire(&ptable.lock);
highest = 0;
// Cari proses RUNNABLE dengan prioritas tertinggi
for(p = ptable.proc; p < &ptable.proc[NPROC]; p++){
if(p->state != RUNNABLE)
;
if(highest == 0 || p->priority < highest->priority)
highest = p;
}
if(highest != 0){
p = highest;
proc = p;
switchuvm(p);
p->state = RUNNING;
swtch(&cpu->scheduler, proc->context);
switchkvm();
proc = 0;
}
release(&ptable.lock);
}
}
// Enter scheduler. Must hold only ptable.lock
// and have changed proc->state. Saves and restores
// intena because intena is a property of this
// kernel thread, not this CPU. It should
// be proc->intena and proc->ncli, but that would
// break in the few places where a lock is held but
// there's no process.
^G Help ^O Write Out ^W Where Is ^K Cut ^T Execute ^C Location