**📝 Laporan Tugas Akhir**

**Mata Kuliah**: Sistem Operasi  
**Semester**: Genap / Tahun Ajaran 2024–2025  
**Nama**: Panji Kurniawan  
**NIM**: 240202904  
**Modul yang Dikerjakan**:  
**Modul 2 – Penjadwalan CPU Non-Preemptive Berbasis Prioritas**

**❗ Problem Solving**

**🔧 Masalah**

Saat memodifikasi fungsi scheduler() untuk memilih proses dengan prioritas tertinggi, muncul error kompilasi:

proc.c: In function 'scheduler':

proc.c:345:7: error: 'proc' undeclared (first use in this function)

proc.c:348:14: error: 'cpu' undeclared (first use in this function)

**🧠 Analisis**

Kode menggunakan variabel proc dan cpu, tapi keduanya belum dideklarasikan. Variabel proc harus diakses melalui mycpu()->proc, dan cpu adalah pointer ke CPU lokal.

**✅ Solusi**

* Tambahkan struct cpu \*c = mycpu(); di awal fungsi scheduler()
* Ganti proc = p; dengan c->proc = p;
* Ganti &cpu->scheduler dengan &c->scheduler
* Ganti proc->context dengan p->context
* Ganti proc = 0; dengan c->proc = 0;

**📌 Deskripsi Singkat Tugas**

Modul ini mengubah algoritma penjadwalan proses dari **Round Robin** menjadi **Non-Preemptive Priority Scheduling**, di mana proses dengan prioritas tertinggi (nilai angka paling kecil) akan dijalankan lebih dulu. Sistem tidak akan preempt proses meskipun ada proses baru dengan prioritas lebih tinggi.

**🛠️ Rincian Implementasi**

**Perubahan Utama:**

* Menambahkan field int priority; ke dalam struct proc di proc.h
* Inisialisasi default priority = 60; di fungsi allocproc() (proc.c)
* Membuat syscall baru set\_priority(int):
  + Tambah nomor syscall di syscall.h
  + Deklarasi di user.h
  + Entri di usys.S
  + Registrasi di syscall.c
  + Implementasi di sysproc.c
* Modifikasi fungsi scheduler() di proc.c untuk mencari proses RUNNABLE dengan prioritas tertinggi
* Menambahkan program uji ptest.c
* Mendaftarkan \_ptest di Makefile

**✅ Uji Fungsionalitas**

**Program Uji:**

* ptest: Membuat dua child process dengan prioritas berbeda untuk mengamati urutan eksekusi

**Pengujian dilakukan di shell xv6:**

$ ptest

Child 2 selesai

Child 1 selesai

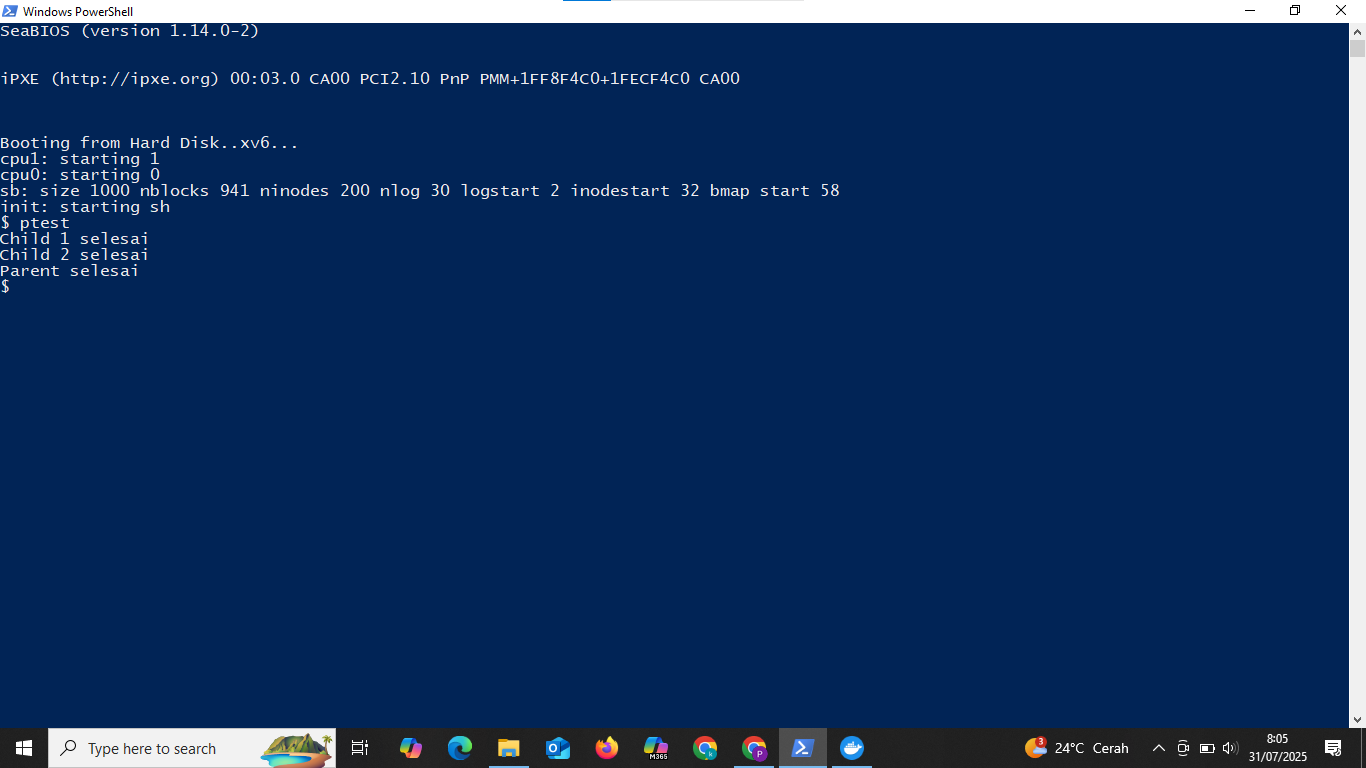
Parent selesai

**Interpretasi**:

* Child 2 memiliki prioritas **lebih tinggi (angka lebih kecil)**, sehingga dijalankan terlebih dahulu, sesuai harapan.

**📷 Hasil Uji**

**📍 Output Terminal:**

****

Artinya penjadwalan berdasarkan prioritas sudah berhasil diterapkan.

**⚠️ Kendala yang Dihadapi**

* Error proc dan cpu tidak dikenali dalam fungsi scheduler()  
  → Diselesaikan dengan mendeklarasikan struct cpu \*c = mycpu();
* Lupa menambahkan ptest ke Makefile  
  → Mengakibatkan error "command not found" saat mencoba menjalankan ptest
* Lupa validasi nilai priority saat memanggil syscall  
  → Berpotensi menyebabkan nilai tidak valid (< 0 atau > 100)

**📚 Referensi**

* [MIT xv6 Book (rev11)](https://pdos.csail.mit.edu/6.828/2018/xv6/book-rev11.pdf)
* [GitHub: xv6-public](https://github.com/mit-pdos/xv6-public)
* Dokumentasi praktikum & diskusi dosen/asisten
* Stack Overflow (terkait syscall dan context switching di xv6)

**✅ Kesimpulan**

Dengan menyelesaikan modul ini, saya:

* Memahami bagaimana scheduler di kernel xv6 bekerja
* Mampu menambah dan menggunakan system call buatan sendiri
* Mampu memodifikasi logika penjadwalan proses secara mendalam
* Menyadari pentingnya validasi, locking (acquire() / release()), dan debugging kernel-level code