CSIM602155: Tugas #2
PSOSh: Shell Pengantar Sistem Operasi

Kuliah Pengantar Sistem Operasi
Semester Gasal 2025/2026

Heri Kurniawan

Due date: Sabtu, 4 Oktober, Pukul: 23.59 WIB

1 Tujuan

- Mahasiswa mampu membuat proses dengan API (`fork`) dan mengganti image proses (`execvp/execlp`) dalam bahasa pemrograman C.
- Mahasiswa mampu mengimplementasikan komunikasi antar proses dengan menggunakan metode message passing `pipe`
- Mahasiswa mampu membuat shell interpreter sederhana (PSOSh) yang dapat mengeksekusi perintah user
- Mahasiswa mampu membuat perintah linux `status` yang dapat dieksekusi pada shell PSOSh

2 Deskripsi

Setelah lama berinteraksi dengan Bash shell, Anda mempunyai ide untuk membuat shell sendiri yang bernama **PSOSh** dengan spesifikasi sebagai berikut:

1. **Mampu menjalankan perintah umum linux**

   (a) Perintah tanpa pipe
   Pada shell PSOSh, User dapat menjalankan perintah umum linux tanpa pipe seperti `ls -al`, `mkdir`, `touch` dan lainnya dengan mengimplementasikan `fork` dan `execvp/execlp` (pilih salah satu, pakai `execvp` atau `execlp`).

   (b) Perintah dengan pipe
   User dapat menjalankan perintah umum linux dengan pipe buatan sendiri "`:`" yang Anda implementasikan dengan API pipe. Jika pada shell Bash, User menggunakan pipe `|` untuk operasi seperti `ls -al | wc -al`, `ls | grep .txt`, dan lainnya, maka pada shell PSOsh, User menggunakan pipe `:` untuk operasi `ls -al : wc -al` atau `ls : grep .txt`. Mengingat `ls`, `mkdir` dan lainnnya merupakan perintah eksternal, maka gunakan fungsi `dup2()` untuk menghubungkan command1 dan command2 ke file descriptor (fd) pipe. Dengan cara ini, output command1 yang kemonitor (stdout) dialihkan ke pipe fd[1] dan command2 menerima input (stdin) dari pipe fd[0].

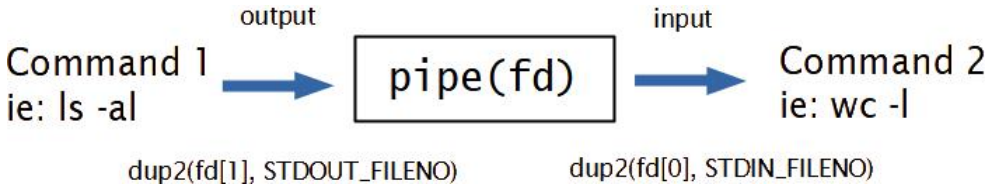

Figure 1: pipe

File Descriptor adalah index dari tabel *open file* yang sedang dibuka oleh proses. Jika proses membuka file, maka proses akan merujuk ke file tersebut dalam bentuk indeks integer. Setiap proses minimum mempunyai 3 file descriptor (fd):

Indeks FD	Constant POSIX	Nama	Penggunaan
0	`STDIN_FILENO`	`stdin`	Standard input (keyboard)
1	`STDOUT_FILENO`	`stdout`	Standard output (monitor)
2	`STDERR_FILENO`	`stderr`	Standard error (monitor, unbuffered)

Table 1: *File descriptor* standar pada sistem Linux/Unix

2. **Mampu mengeksekusi perintah** `status`

   Seperti halnya `ls`, `mkdir`, dan lainnya, Anda mempunyai ide untuk membuat perintah sendiri yang bernama `status`. Program status dibuat dalam bahasa C dan dapat dieksekusi melalui shell dengan parameter sebagai berikut:

   (a) `status -c`
   Memberikan informasi CPU.
   Perintah linux yang dieksekusi pada kode program:
   `lscpu | egrep 'Arch|Model|Core|Thread|Byte|L1d|L1i|L2|L3|Address'`

   (b) `status -p`
   Memberikan informasi top 5 proses yang mengkonsumsi memori.
   Perintah linux yang dieksekusi pada kode program:
   `ps -eo user,cmd,%cpu,%mem --sort=-%mem | head -n 6`

   (c) `status -m`
   Memberikan informasi memori.
   Perintah linux yang dieksekusi pada kode program:
   `free -h`

   (d) `status -d`
   Memberikan informasi disk.
   Perintah linux yang dieksekusi pada kode program:
   `df -h /`

   (e) `status -a`
   Memberikan informasi gabungan poin (a), (b), (c) dan (d).

   Penggunaan: melalui shell PSOSh, user dapat mengetikkan `status -c` atau dengan lebih dari satu parameter seperti `status -c -p` atau `status -c -p -m -d` . Program `status` berperan sebagai *wrapper* dari perintah perintah linux diatas.

3. **Struktur Program**

   (a) Penempatan file binary hasil kompilasi
   Berdasarkan poin 1 dan 2, Anda membuat dua program `t2-psosh.c` (program shell) dan `status.c` (program perintah). Hasil kompilasi `t2-psosh.c` akan berada pada direktori yang sama sedangkan file binary `status.c` akan berada pada `$HOME/bin`, dimana `$HOME` adalah home directory Anda. Jika `$HOME` user adalah `/home/rudi`, maka file `status` akan berada difolder `/home/rudi/bin`. Untuk mempermudah, Anda akan diberikan file `Makefile` yang secara otomatis mengimplementasikan ketentuan diatas.

   (b) Path eksekusi
   Secara default API `execvp/execlp` akan mengeksekusi file binary yang terdapat pada folder `$PATH`. Jika Anda eksekusi `$ echo $PATH` maka sistem akan memberikan lokasi folder `/usr/local/sbin:/usr/local/bin:/usr/sbin:/usr/bin:/sbin:/bin:/usr/games: /usr/local/games:/snap/bin`. Agar program binary `status` dieksekusi secara otomatis, tambahkan path $HOME/bin pada $PATH secara permanen:

   ```
   $ echo 'export PATH="$PATH:$HOME/bin"' >> ~/.bashrc
   $ source ~/.bashrc
   ```

4. **(BONUS 10 Poin) Mampu menyimpan *history* rekaman perintah sebelumnya**

   Seperti Bash, Shell PSOSh dapat menampilkan perintah sebelumnya jika user menekan tombol panah keatas (↑) dan panah kebawah (↓) untuk perintah berikutnya. (Hint: gunakan library `readline`)

Implementasikan pekerjaan diatas pada mesin virtual sistem operasi Linux atau Windows Subsystem for Linux (WSL). Berikan komentar pada kode program untuk memperjelas cara kerja program Anda. Untuk mendapatkan nilai maksimal 110 poin dan mempermudah penilaian, output pekerjaan Anda HARUS sama seperti pada tampilan program berikut: `https://youtu.be/rcUHVq9nTkQ`.

Saran untuk mempercepat pengerjaan tugas:

- Pahami alur logik program.
- Pahami cara membuat anak proses dengan menggunakan API `fork` dan menggantinya dengan image perintah (ie: `ls`, `status`, `mkdir`, dan lainnya) melalui fungsi `execvp` atau `execlp`.
- Pahami apa itu file descriptor, cara kerja `pipe` dan `dup2`
- Lakukan teknik *divide and conquer* untuk mempermudah pengembangan program.
- Mulailah mengerjakan spesifikasi yang paling mudah
- Kerjakan tugas sejak dini karena Anda tidak tahu masalah apa yang akan Anda hadapi kedepannya.
- Gunakan google jika menemui permasalahan. Google adalah teman Anda :)

3 Pengumpulan

1. Buat video yang menjelaskan kode dan demo jalannya program pada youtube dengan mode **unlisted** (tidak publik). Pada video, presenter dapat menampakkan wajah ataupun hanya audio suara tanpa wajah.
2. File yang dikumpul:
   - `t2-psosh` - kode program shell psosh dan komentar
   - `status.c` - kode program status dan komentar
   - `readme.txt` - berisikan informasi **link youtube**, **Nama-NPM**, dan cara menjalankan program.
   - file.txt - untuk testing pipe
   - `Makefile` - untuk kompilasi dengan make
3. Zip kelima file diatas dan upload via Scele dengan penamaan: **t2_kelas(A/B/C)_npm_nama.zip**

Hindari tindakan PLAGIARISME. Terbukti melakukan plagiarisme akan mendapatkan SANKSI maksimal nilai E.

Selamat Mengerjakan