**Soal Shift Modul 3**

**Sistem Operasi 2021**

1. Waktu pengerjaan dimulai Selasa (4/5) pukul 10.00 WIB hingga Sabtu (8/5) pukul 22.00 WIB.
2. Praktikan diharapkan membuat laporan penjelasan dan penyelesaian soal dalam bentuk Readme(github) untuk laporan resmi.
3. Format nama repository github “soal-shift-sisop-modul-Y-XXX-2021” (contoh:soal-shift-sisop-modul-1-A01-2021).
4. Struktur repository seperti berikut:

---soal1:

---Client

---client.c

---Server

---server.c

---soal2:

---soal2a.c

---soal2b.c

---soal2c.c

---soal3:

---soal3.c

Jika **melanggar struktur repo** akan **dianggap sama dengan curang** dan menerima konsekuensi **sama** dengan **melakukan kecurangan**.

1. Setelah pengerjaan selesai, semua script bash, awk, dan file yang berisi cron job ditaruh di github masing - masing kelompok, dan link github diletakkan pada form yang disediakan.
2. Commit terakhir maksimal 10 menit setelah waktu pengerjaan berakhir. Jika melewati maka akan dinilai berdasarkan commit terakhir.
3. Jika tidak ada pengumuman perubahan soal oleh asisten, maka soal dianggap dapat diselesaikan.
4. Jika ditemukan soal yang tidak dapat diselesaikan, harap menuliskannya pada Readme beserta permasalahan yang ditemukan.
5. Praktikan tidak diperbolehkan menanyakan jawaban dari soal yang diberikan kepada asisten maupun praktikan dari kelompok lainnya.
6. Jika ditemukan **indikasi kecurangan** dalam bentuk apapun di pengerjaan soal shift, maka **nilai dianggap 0**.
7. Pengerjaan soal shift sesuai dengan modul yang telah diajarkan.
8. Zip dari repository dikirim ke email asisten penguji dengan subjek yang sama dengan nama judul repository, dikirim sebelum deadline dari soal shift
9. Keverk adalah orang yang cukup ambisius dan terkenal di angkatannya. Sebelum dia menjadi ketua departemen di HMTC, dia pernah mengerjakan suatu proyek dimana keverk tersebut meminta untuk membuat server database buku. Proyek ini diminta agar dapat digunakan oleh pemilik aplikasi dan diharapkan bantuannya dari pengguna aplikasi ini.

Di dalam proyek itu, Keverk diminta:

1. Pada saat client tersambung dengan server, terdapat dua pilihan pertama, yaitu register dan login. Jika memilih register, client akan diminta input id dan passwordnya untuk dikirimkan ke server. User juga dapat melakukan login. Login berhasil jika id dan password yang dikirim dari aplikasi client sesuai dengan list akun yang ada didalam aplikasi server. Sistem ini juga dapat menerima multi-connections. Koneksi terhitung ketika aplikasi client tersambung dengan server. Jika terdapat 2 koneksi atau lebih maka harus menunggu sampai client pertama keluar untuk bisa melakukan login dan mengakses aplikasinya. Keverk menginginkan lokasi penyimpanan id dan password pada file bernama **akun.txt** dengan format :

**akun.txt**

|  |
| --- |
| id:password  id2:password2 |

1. Sistem memiliki sebuah database yang bernama **files.tsv.** Isi dari **files.tsv** ini adalah **path file saat berada di server**, **publisher**, dan **tahun publikasi**. Setiap penambahan dan penghapusan file pada folder file yang bernama **FILES** pada server akan memengaruhi isi dari **files.tsv.** Folder **FILES** otomatis dibuat saat server dijalankan.
2. Tidak hanya itu, Keverk juga diminta membuat fitur agar client dapat menambah file baru ke dalam server. Direktori **FILES** memiliki struktur direktori di bawah ini :

Direktori **FILES**

|  |
| --- |
| File1.ekstensi  File2.ekstensi |

Pertama client mengirimkan input ke server dengan struktur sebagai berikut :

Contoh Command Client :

|  |
| --- |
| add |

Output Client Console:

|  |
| --- |
| Publisher:  Tahun Publikasi:  Filepath: |

Kemudian, dari aplikasi client akan dimasukan data buku tersebut (perlu diingat bahwa Filepath ini merupakan **path file yang akan dikirim ke server**). Lalu client nanti akan melakukan pengiriman file ke aplikasi server dengan menggunakan socket. Ketika file diterima di server, maka row dari files.tsv akan bertambah sesuai dengan data terbaru yang ditambahkan.

1. Dan client dapat mendownload file yang telah ada dalam folder FILES di server, sehingga sistem harus dapat mengirim file ke client. Server **harus** melihat dari files.tsv untuk melakukan pengecekan apakah file tersebut valid. Jika tidak valid, maka mengirimkan pesan error balik ke client. Jika berhasil, file akan dikirim dan akan diterima ke client di **folder client** tersebut.

Contoh Command client

|  |
| --- |
| download TEMPfile.pdf |

1. Setelah itu, client juga dapat menghapus file yang tersimpan di server. Akan tetapi, Keverk takut file yang dibuang adalah file yang penting, maka file hanya akan diganti namanya menjadi ‘old-NamaFile.ekstensi’. Ketika file telah diubah namanya, maka row dari file tersebut di file.tsv akan terhapus.

Contoh Command Client:

|  |
| --- |
| delete TEMPfile.pdf |

1. Client dapat melihat semua isi **files.tsv** dengan memanggil suatu perintah yang bernama see.Output dari perintah tersebut keluar dengan format.

Contoh Command Client :

|  |
| --- |
| see |

Contoh Format Output pada Client:

|  |
| --- |
| Nama:  Publisher:  Tahun publishing:  Ekstensi File :  Filepath :  Nama:  Publisher:  Tahun publishing:  Ekstensi File :  Filepath : |

1. Aplikasi client juga dapat melakukan pencarian dengan memberikan suatu string. Hasilnya adalah semua nama file yang mengandung string tersebut. Format output seperti nomor 6.

Contoh Client Command:

|  |
| --- |
| find TEMP |

1. Dikarenakan Keverk waspada dengan pertambahan dan penghapusan file di server, maka Keverk membuat suatu log untuk server yang bernama **running.log**. Contoh isi dari log ini adalah

**running.log**

|  |
| --- |
| Tambah : File1.ektensi (id:pass)  Hapus : File2.ektensi (id:pass) |

Akan tetapi, kevek saat itu masih belum paham dengan sistem server dan client. Bantulah Keverk untuk menyelesaikan proyek ini!

Note:

* Dilarang menggunakan system() dan execv(). Silahkan dikerjakan sepenuhnya dengan thread dan socket programming.
* Untuk download dan upload silahkan menggunakan file teks dengan ekstensi dan isi bebas (yang ada isinya bukan touch saja dan tidak kosong) dan requirement untuk benar adalah percobaan dengan minimum 5 data.

Struktur Direktori:

|  |
| --- |
| .  ├── Client  │ ├── client.c  │ ├── File2.extensi  │ └── File1.extensi  └── Server  ├── akun.txt  ├── files.tsv  ├── server.c  ├── running.log  └── FILES  ├── File2.extensi  └── File1.ekstensi |

1. Crypto (kamu) adalah teman Loba. Suatu pagi, Crypto melihat Loba yang sedang kewalahan mengerjakan tugas dari bosnya. Karena Crypto adalah orang yang sangat menyukai tantangan, dia ingin membantu Loba mengerjakan tugasnya. Detil dari tugas tersebut adalah:
   1. Membuat program perkalian matrix (4x3 dengan 3x6) dan menampilkan hasilnya. Matriks nantinya akan berisi angka 1-20 (tidak perlu dibuat filter angka).
   2. Membuat program dengan menggunakan matriks output dari program sebelumnya (program soal2a.c) (**Catatan!:** gunakan shared memory). Kemudian matriks tersebut akan dilakukan perhitungan dengan matrix baru sebagai berikut contoh perhitungan untuk matriks yang ada. Perhitungannya adalah setiap cel yang berasal dari matriks A menjadi angka untuk faktorial, lalu cel dari matriks B menjadi batas maksimal faktorialnya (dari paling besar ke paling kecil) (**Catatan!:** gunakan thread untuk perhitungan di setiap cel).

**Ketentuan**

|  |
| --- |
| If a >= b -> a!/(a-b)!  If b > a -> a!  If 0 -> 0 |

Contoh :

|  |  |  |
| --- | --- | --- |
| **A** | **B** | **Angka-Angka Faktorial** |
| 4 | 4 | 4 3 2 1 |
| 4 | 5 | 4 3 2 1 |
| 4 | 3 | 4 3 2 |
| 4 | 0 | 0 |
| 0 | 4 | 0 |
| 4 | 6 | 4 3 2 1 |

Contoh :

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
| Matriks A | | Matriks B | | Matriks Hasil | |
| 0 | 4 | 0 | 4 | 0 | 4\*3\*2\*1 |
| 4 | 5 | 6 | 2 | 4\*3\*2\*1 | 5\*4 |
| 5 | 6 | 6 | 0 | 5\*4\*3\*2\*1 | 0 |

* 1. Karena takut lag dalam pengerjaannya membantu Loba, Crypto juga membuat program (soal2c.c) untuk mengecek 5 proses teratas apa saja yang memakan resource komputernya dengan command “ps aux | sort -nrk 3,3 | head -5” (**Catatan!:** Harus menggunakan IPC Pipes)

Note:

* Semua matriks berasal dari input ke program.
* Dilarang menggunakan system()

1. Seorang mahasiswa bernama Alex sedang mengalami masa gabut. Di saat masa gabutnya, ia memikirkan untuk merapikan sejumlah file yang ada di laptopnya. Karena jumlah filenya terlalu banyak, Alex meminta saran ke Ayub. Ayub menyarankan untuk membuat sebuah program C agar file-file dapat dikategorikan. Program ini akan memindahkan file sesuai ekstensinya ke dalam folder sesuai ekstensinya yang folder hasilnya terdapat di working directory ketika program kategori tersebut dijalankan.

Contoh apabila program dijalankan:

|  |
| --- |
| # Program soal3 terletak di /home/izone/soal3  $ ./soal3 -f path/to/file1.jpg path/to/file2.c path/to/file3.zip  #Hasilnya adalah sebagai berikut  /home/izone  |-jpg  |--file1.jpg  |-c  |--file2.c  |-zip  |--file3.zip |

1. Program menerima opsi -f seperti contoh di atas, jadi pengguna bisa menambahkan argumen file yang bisa dikategorikan sebanyak yang diinginkan oleh pengguna.

Output yang dikeluarkan adalah seperti ini :

|  |
| --- |
| File 1 : Berhasil Dikategorikan (jika berhasil)  File 2 : Sad, gagal :( (jika gagal)  File 3 : Berhasil Dikategorikan |

1. Program juga dapat menerima opsi -d untuk melakukan pengkategorian pada suatu directory. Namun pada opsi -d ini, user hanya bisa memasukkan input 1 directory saja, tidak seperti file yang bebas menginput file sebanyak mungkin. Contohnya adalah seperti ini:

|  |
| --- |
| $ ./soal3 -d /path/to/directory/ |

Perintah di atas akan mengkategorikan file di /path/to/directory, lalu hasilnya akan disimpan di working directory dimana program C tersebut berjalan (hasil kategori filenya bukan di /path/to/directory).

Output yang dikeluarkan adalah seperti ini :

|  |
| --- |
| Jika berhasil, print “Direktori sukses disimpan!”  Jika gagal, print “Yah, gagal disimpan :(“ |

1. Selain menerima opsi-opsi di atas, program ini menerima opsi \*, contohnya ada di bawah ini:

|  |
| --- |
| $ ./soal3 \\* |

Opsi ini akan mengkategorikan seluruh file yang ada di working directory ketika menjalankan program C tersebut.

1. Semua file harus berada di dalam folder, jika terdapat file yang tidak memiliki ekstensi, file disimpan dalam folder “Unknown”. Jika file hidden, masuk folder “Hidden”.
2. Setiap 1 file yang dikategorikan dioperasikan oleh 1 thread agar bisa berjalan secara paralel sehingga proses kategori bisa berjalan lebih cepat.

Namun karena Ayub adalah orang yang hanya bisa memberi ide saja, tidak mau bantuin buat bikin programnya, Ayub meminta bantuanmu untuk membuatkan programnya. Bantulah agar program dapat berjalan!

Catatan:

* Kategori folder tidak dibuat secara manual, harus melalui program C
* Program ini tidak case sensitive. Contoh: JPG dan jpg adalah sama
* Jika ekstensi lebih dari satu (contoh “.tar.gz”) maka akan masuk ke folder dengan titik terdepan (contoh “tar.gz”)
* Dilarang juga menggunakan fork-exec dan system()
* Bagian b dan c berlaku **rekursif**