Nama : Mahardika Krisna Ihsani

Kelas : A

NPM : 1806141284

Weekly Log 2 System Programming

Things That I've Learned

Pada minggu ini saya mempelajari bahwa unix system menggunakan concept of universality pada implementasi I/O. Ini artinya, bahwa segala file operation tidak hanya berlaku pada file dan folder saja, tetapi operasi tersebut juga berlaku pada device seperti read, open, dll. Selain itu, Saya juga mempelajari bahwa program dengan process mempunyai keterkaitan. Program sendiri dapat terdiri dari beberapa process. Misalnya program image viewer mempunyai beberapa process seperti process untuk melihat gambar, process melakukan editing, dsb. Dan juga, susunan memory secara garis besar terdiri atas environment, stack & heap, BSS, static data, dan text segment.

Pada minggu ini juga dibahas mengenai perbedaan antara static libraries dengan shared libraries. Perbedaan keduanya adalah, static libraries mengkopi source code ke dalam source code program yang menggunakan library tersebut sedangkan shared libraries tidak mengkopi source code ke dalam source code program yang menggunakannya.

Pada linux, terdapat direktori /proc. Direktori ini merupakan filesystem yang berguna untuk menampilkan informasi-informasi yang berkaitan dengan kernel, seperti informasi hardware, process, uptime, dll.

System call ini berguna karena dalam apabila aplikasi ingin mengakses suatu device, tidak bisa aplikasi tersebut mengakses langsung device tersebut atau mencari device tersebut sendiri. Aplikasi tersebut tentunya memerlukan instruksi yang ada di kernel mode karena kernel mode mempunyai akses penuh terhadap device-device yang terhubung pada komputer. Oleh karena itu system call berguna di sini sebagai penghubung antara aplikasi dengan instruksi yang memerlukan kernel access.

Pada minggu sebelumnya dibahas bahwa suatu sistem terdiri atas dua mode, yaitu user mode dan kernel mode. Sudah pernah disinggung sebelumnya, bahwa system call berfungsi sebagai

jembatan antara user mode dengan kernel mode. Pada minggu ini, dibahas bagaimana system call bekerja menghubungkan antara user mode dengan kernel mode. Ketika suatu aplikasi di user mode ingin melakukan eksekusi yang membutuhkan kernel mode, maka aplikasi tersebut akan mendegelasikan perintah tersebut melalui wrapper function. Wrapper function akan menginisiasi fungsi apa saja yang perlu dieksekusi pada kernel mode dan mempersiapkan resources yang diperlukan seperti register dan memory. Lalu pada kernel mode terdapat trap handler. Trap handler ini akan melayani fungsi-fungsi yang dibutuhkan melalui wrapper function dan melakukan switching dari user mode ke kernel mode. Fungsi-fungsi tersebut akan disalurkan ke system call service routine untuk dieksekusi. Eksekusi tersebut akan menghasilkan suatu output atau error. Output/error akan disalurkan kembali ke user mode melalui trap handler. Trap handler akan melalui switching ke user mode dan dari situ akan disalurkan ke register. Dari register menuju ke user stack sehingga hasil/error dari kernel mode bisa tersalurkan ke user mode.

Disebutkan sebelumnya bahwa system call bisa saja menghasilkan error. Biasanya system call akan mengeluarkan -1 untuk error dan bilangan positif untuk instruksi yang berhasil dieksekusi. Error ini biasanya disertakan dengan errno yang menyatakan ID dari tipe error yang dihasilkan. Untuk menentukan ID serta jenis error, kita bisa menggunakan command errno -ls. Walaupun begitu beberapa system call akan mengeluarkan -1 walaupun proses tersebut berhasil berhenti. Untuk mengecek apakah system call tersebut benar-benar berhenti, kita bisa menginclude library <errno.h> pada aplikasi dan mengeset errno=0. Setelah pemanggilan system call, kita perlu mengecek kembali nilai errno. Apabila errno-nya tidak sama dengan 0, maka benar terjadi error dalam system call tersebut. Secara default, semua system call atau library function mengeset nilai errno tidak sama dengan 0.

Dalam pengaksesan system call, tentunya lebih baik kita menggunakan library function. Library function ini adalah fungsi-fungsi siap pakai yang terkandung dalam suatu library. System call lebih baik dieksekusi melalui fungsi-fungsi tersebut karena fungsi tersebut merupakan wrapper dari beberapa system call sehingga dapat menghemat waktu untuk melakukan operasi kernel yang sekiranya sederhana seperti melihat file, menyimpan file, dsb. Selain itu, apabila system call dipanggil langsung, tentunya akan lebih riskan terhadap kerusakan sistem dibandingkan bila dipanggil melalui library function karena library function ini sudah teruji dan *robust*.

Something That's Still Unclear For Me

Saya masih tidak paham mengapa system call secara default mengeset nilai errno=0. Padahal apabila diset ke errno tersebut, dapat dipastikan apakah system call yang dipanggil benar-benar menghasilkan error atau tidak.

Referensi

- https://youtu.be/f_ANXf2OlCA - - Systems Programming - 02-01/02- System Call - CSCM603127
- https://youtu.be/F1rlFf9HsIo - Systems Programming - 02-02/02- System Call - CSCM603127
- https://man7.org/linux/man-pages/