Week 4 Logbook

Aktivitas Minggu ini
Pada minggu ini, saya melakukan hal-
hal di bawah ini untuk meningkatkan pengetahuan dan pengalaman saya da
lam System Programming:
1. Mengerjakan Pre Test 4
2. Mengerjakan WS4
3. Menonton video materi di Youtube
4. Menghadiri kelas Sysprog pada hari Jum'at tanggal 9 Oktober 2020
5. Mencari sumber referensi tentang process, thread, program, copy-on-
write, beberapa command di Linux, zombie process, orphaned process, da
emon, state suatu process, environment variable, dan cara mengatur env
ironment variable
6. Membaca manual command pada Linux
7. Berdiskusi serta menyamakan pengetahuan dengan teman mengenai mater
i minggu ini
8. Membaca slide yang sudah disiapkan di Scele
9. Trial and error untuk bisa menjalankan command ps dengan berbagai o
psi
10. Trial and error untuk bisa mengubah environment variable secara pe
rmanen di Linux
11. Mencoba-coba command yang dicontohkan di video modul pembelajaran

Thread
Thread merupakan segmen kecil dari suatu proses. Thread juga merupakan
 entitas yang aktif karena merupakan bagian dari proses. Thread cender
ung berumur lebih pendek dari proses dan membutuhkan resource lebih se
dikit ketimbang proses.

Process
Process adalah instance dari program yang sedang dijalankan pada siste
m operasi. Process juga merupakan merupakan entitas yang aktif karena
dijalankan pada memori dinamis. Process biasanya berumur pendek karena
 proses selalu hilang setelah task yang dikerjakan selesai. Selain itu
, process juga membutuhkan resource yang banyak sebab proses membutuhk
an kernel agar dapat mengakses I/O, CPU, memory management, task manag
ement dan lain-lain.

Program
Program adalah kumpulan suatu instruksi yang berfungsi untuk menyelesaikan suatu task. Program juga merupakan sebuah entitas yang pasif karena berada pada memory static atau secondary memory. Program biasanya berumur panjang sebab letaknya hanya ada di satu tempat dan tidak akan hilang sampai user menghapusnya. Program juga tidak membutuhkan resource karena hanya membutuhkan space untuk mengeksekusi instruksi-instruksi yang berada di dalamnya

Zombie Process
Zombie process adalah suatu proses yang sudah selesai eksekusinya atau sudah di-
terminasi, namun proses tersebut masih tersimpan pada process table sehingga memakan resource yang ada. Biasanya zombie process ini terjadi akibat pembuatan program yang kurang baik dan kurang berkualitas.

Orphaned Process
Orphaned process adalah suatu proses yang parent process nya sudah selesai atau sudah di-
terminasi. Biasanya orphaned process ini diadopsi oleh PID 1, yaitu proses init.

Daemon Process
Daemon process adalah suatu proses yang sengaja tidak memiliki parent process pada pembuatannya agar bisa dijadikan sebagai background process. Biasanya daemon process hanya mewariskan pid proses init, namun init bukanlah parent dari daemon. Daemon juga tidak dapat dikontrol oleh pengguna lewat terminal sehingga biasanya daemon tidak memiliki suatu file descriptor.

Environment Variable
Environment variable adalah suatu variabel dinamis yang diatur diluar jalannya suatu program sehingga bisa berpengaruh untuk bagaimana suatu proses bisa berjalan di suatu sistem operasi, khususnya kernel. Environment variable ini biasanya berupa format nama variabel dan value dari variabel tersebut.

Copy-on-write

Copy-on-
write adalah sebuah teknik dalam mengatur resource yang dimiliki oleh suatu sistem dimana teknik ini memungkinkan kita untuk menduplikasi suatu resource yang dapat dimodifikasi tanpa harus membuat file-
nya setelah proses menyalin dilakukan.

Perbedaan _exit() dengan exit()
exit() ikut membuang I/O buffer dan menjalankan beberapa fungsi yang ada pata atexit() ketika melakukan penutupan suatu proses. Sedangkan _exit() cukup straightforward sebab hanya sekadar menutup proses tanpa membuang buffer-buffer yang ada.

Kegunaan PID dan PPID serta Cara Mendapatkannya
PID adalah ID dari suatu proses, dan PPID adalah ID orang tua proses dari suatu proses. Cara mendapatkan PID biasanya adalah dengan menggunakan system call / command getpid(). Sedangkan cara mendapatkan PPID adalah dengan menggunakan system call / command getppid(). Adapun keguna
an dari PID adalah sebagai identifier unik agar sistem dan pengguna bisa membeda-
bedakan proses apa saja yang sedang berjalan di sistem. Sedangkan PPID berguna untuk mengetahui ancestor dari suatu proses. Selain itu, PPID juga bisa berguna untuk membunuh suatu Zombie Process agar tidak lagi memakan resource dari sistem.

Apa yang Terjadi ketika fork Dieksekusi pada Suatu Proses
Ketika kita mengeksekusi fork() pada suatu proses yang sedang berjalan, sistem akan membuat sebuah proses baru yang merupakan duplikat dari proses tersebut. Proses baru ini memiliki PPID yang merujuk pada proses yang dijadikan argumen perintah fork().

Copy-on-write pada System Call fork
Ketika fork dijalankan pada suatu proses, pages dari parent process tidak akan di copy untuk child process melainkan pages-
nya akan dibagikan dengan child dan parent process. Setiap kali ada perubahan yang dilakukan oleh salah satu proses, bagian kecil dari page yang berubah akan dikonstruksi untuk proses yang melakukan perubahan tersebut sehingga proses tersebut akan menggunakan salinan yang telah dibuat dan tidak lagi bersama-
sama menggunakan page yang sama dengan proses lainnya.

Sumber Bacaan Saya pada Week-4 untuk Logbook, Worksheet, dan PreTest

https://www.geeksforgeeks.org/getppid-getpid-linux/
https://www.unix.com/programming/116721-difference-between-exit-_exit.html
http://www.linfo.org/pid.html
https://delightlylinux.wordpress.com/2012/06/25/what-is-pid-and-ppid
https://en.wikipedia.org/wiki/Copy-on-write
https://unix.stackexchange.com/questions/58145/how-does-copy-on-write-in-fork-handle-multiple-fork
https://www.gmarik.info/blog/2012/orphan-vs-zombie-vs-daemon-processes
https://man7.org/linux/man-pages/man1/ps.1.html
https://stackoverflow.com/questions/16944886/how-to-kill-zombie-process
https://man7.org/linux/man-pages/man1/top.1.html
https://www.geeksforgeeks.org/states-of-a-process-in-operating-systems/
https://www.javatpoint.com/os-process-states
https://www.tecmint.com/ps-command-examples-for-linux-process-monitoring/
https://unix.stackexchange.com/questions/117467/how-to-permanently-set-environmental-variables
https://unix.stackexchange.com/questions/16883/what-is-the-maximum-value-of-the-process-id
https://unix.stackexchange.com/questions/124040/how-to-determine-the-max-user-process-value
https://serverfault.com/questions/489597/what-happens-when-pid-max-is-reached
http://www.cs.kent.edu/~dostanle/courses/cs43203/notes/daemons/
https://medium.com/chingu/an-introduction-to-environment-variables-and-how-to-use-them-f602f66d15fa
https://en.wikipedia.org/wiki/Environment_variable