Nama: Irsyad Nafi Alif Perdana

NIM: 33121101067

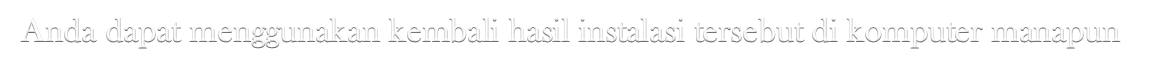
Kelas: IF2C Pagi

**MODULE 11: WHERE DATA IS STORED**

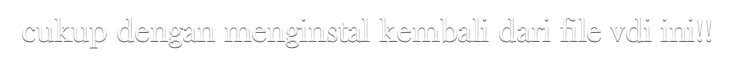
|  |  |
| --- | --- |
| POKOK BAHASAN:  ﻫ Memeriksa file sistem di direktori /proc  ﻫ Penggunaan perintah ps untuk melihat informasi proses pada sistem  ﻫ Mengelola proses mulai dari start, stop, dan ringkasannya  ﻫ Eksplorasi log file  ﻫ Kelola kemampuan untuk memuat librari bersama | OBJEKTIF:   * Mahasiswa mampu mengenal pseudo file system di direktori /proc * Mahasiswa mampu menggunakan serta memanfaatkan perintah ps untuk keperluan monitoring sistem * Mahasiswa mampu mengelola proses yang akan, sedang, dan mengakhiri proses * Mahasiswa mampu mengenal log file dari sistem * Mahasiswa mampu memanggil librari   bersama |

TUGAS PRAKTIKUM:

1. Mengerjakan setiap langkah yang ada di modul praktikum dengan bukti print screen dari output yang didapat dan harus melalui shell dengan nama Anda.
2. Menjawab seluruh soal praktikum
3. Mengumpulkan hasil praktikum di learning pada link yang telah disediakan di akhir sesi praktikum dengan format file docx/pdf, dan **TIDAK ADA PENAMBAHAN WAKTU PENGUMPULAN**
4. Berikan nama file dengan nama Anda



**cukup dengan menginstal kembali dari file vdi ini!!**



PENJELASAN:

Dalam praktikum kali ini, kita akan mengeksplorasi direktori **/proc** dan perintah yang berkomunikasi dengan kernel Linux. Direktori **/proc** tampaknya merupakan direktori biasa,seperti

/usr atau /etc, tetapi sebenarnya tidak. Tidak seperti direktori /usr atau /etc, yang biasanya ditulis ke disk drive, direktori **/proc** adalah sistem file pseudo (pseudo file system) yang disimpandalam memori komputer.



Direktori **/proc** berisi subdirektori untuk setiap proses yang berjalan pada sistem. Program seperti **ps** dan **top** membaca informasi tentang menjalankan proses dari direktori ini. Direktori **/proc** juga berisi informasi tentang sistem operasi dan perangkat kerasnya dalam file seperti **/proc/cpuinfo**,

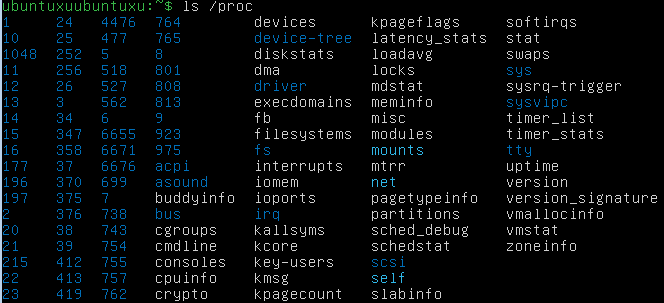
# /proc/meminfo dan /proc /devices.

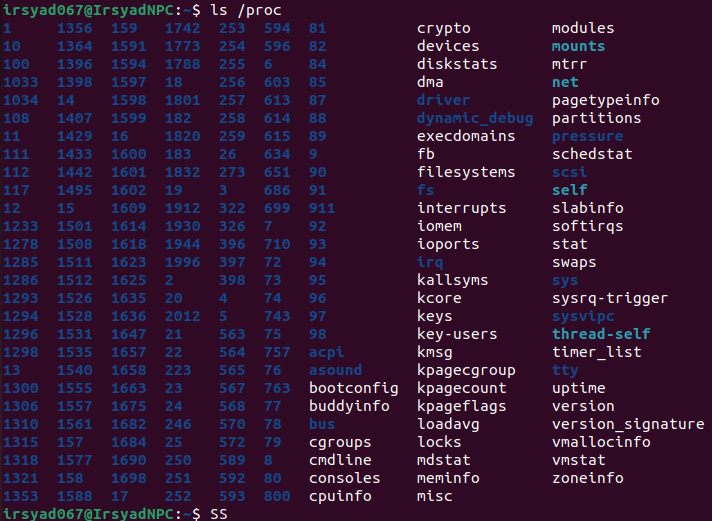
Subdirektori **/proc/sys** berisi file pseudo yang dapat digunakan untuk mengubah pengaturan kernel yang sedang berjalan. Karena file-file ini bukan file "nyata", editor tidak boleh digunakan untuk mengubahnya, sebaliknya kita harus menggunakan perintah **echo** atau **sysctl** untuk menimpa isi file-file ini. Untuk alasan yang sama, jangan mencoba untuk melihat file-file ini di editor, tetapi gunakan perintah **cat** atau **sysctl** sebagai gantinya.

Untuk perubahan konfigurasi permanen, kernel menggunakan file **/etc/sysctl.conf**. Biasanya, file ini digunakan oleh kernel untuk membuat perubahan pada file /proc saat sistem dijalankan.

LANGKAH 1:

Dalam langkah ini, kita akan memeriksa beberapa file yang terdapat di direktori **/proc**. Ketikkan perintah ini di shell: **ls /proc**, maka akan ditampilkan informasi seperti yang terlihat pada gambar 10.1





Gambar 10.1

Ingat bahwa direktori yang memiliki angka untuk nama mewakili proses yang berjalan pada sistem. Proses pertama selalu **/sbin /init**, sehingga direktori **/proc/1** akan berisi file dengan informasi tentang proses init yang sedang berjalan.

File cmdline di dalam direktori proses (**/proc/1/cmdline**, misalnya) akan menampilkan perintah yang dieksekusi. Urutan di mana proses lain dimulai sangat bervariasi dari sistem ke sistem. Karena konten file ini tidak mengandung karakter baris baru, perintah **echo** akan dieksekusi untuk menyebabkan prompt untuk pergi ke baris baru.

LANGKAH 2:

Gunakan **cat** dan kemudian **ps** untuk melihat informasi tentang proses /sbin/init (Process IDentifier (PID) dari 1). Ketikkan perintah ini di shell: **cat /proc/1/cmdline; echo**, maka akan ditampilkan informasi seperti yang terlihat pada gambar 10.2



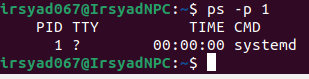


Gambar 10.2

**Catatan:** Perintah echo dalam contoh ini dijalankan segera setelah perintah cat. Karena tidak memiliki argumen, perintah ini berfungsi hanya untuk menempatkan command prompt di baris baru. Jalankan perintah cat tanpa echo untuk melihat perbedaannya.

Ketikkan perintah ini di shell: **ps –p 1**, maka akan ditampilkan informasi seperti yang terlihat pada gambar 10.3





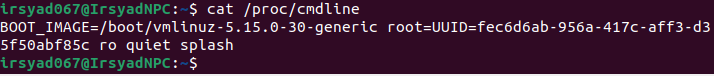
Gambar 10.3

File-file lain di direktori **/proc** berisi informasi tentang sistem operasi. Tugas-tugas berikut akan digunakan untuk melihat dan memodifikasi file-file ini.

LANGKAH 3:

Lihat file /proc/cmdline untuk melihat argumen apa yang diteruskan ke kernel pada saat boot. Ketikkan perintah ini di shell: **cat /proc/cmdline**, maka akan ditampilkan informasi seperti yang terlihat pada gambar 10.4





Gambar 10.4

LANGKAH 4:

Ketikkan perintah ini di shell: **ping localhost > /dev/null**, maka akan ditampilkan informasi seperti yang terlihat pada gambar 10.5





Gambar 10.5

Output dari ping sedang diarahkan ke file **/dev/null** (yang umumnya dikenal sebagai bit bucket).

Perhatikan bahwa terminal nampak menutup dengan perintah ini. Ini karena menjalankan perintah ini di "foreground". Sistem akan terus melakukan ping hingga proses dihentikan atau ditangguhkan oleh pengguna.

Hentikan proses foreground dengan menekan Ctrl-C. LANGKAH 5:

Selanjutnya, untuk memulai proses yang sama di latar belakang (background). Ketikkan perintah

ini di shell: **ping localhost > /dev/null &**, maka akan ditampilkan informasi seperti yang terlihat pada gambar 10.6



Gambar 10.6

Dengan menambahkan tanda & ke akhir perintah, proses dimulai di latar belakang yang memungkinkan pengguna untuk mempertahankan kontrol terminal.

Cara yang lebih mudah untuk memasukkan perintah di atas adalah dengan memanfaatkan perintah history. Kita bisa menekan Tombol Panah Atas ↑ pada keyboard, tambahkan Spasi dan & ke akhir perintah, lalu tekan tombol Enter. Ini adalah penghemat waktu ketika memasukkan perintah serupa.

Perhatikan bahwa perintah sebelumnya mengembalikan informasi berikut: [2] 6738

Ini berarti bahwa proses ini memiliki nomor job/pekerjaan 2 (seperti yang ditunjukkan oleh output [2]) dan ID Proses (PID) 6738. Setiap terminal atau shell akan memiliki nomor pekerjaan yang unik. PID adalah seluruh sistem sementara setiap proses memiliki nomor ID unik.

Informasi ini penting ketika melakukan manipulasi proses tertentu, seperti menghentikan proses atau mengubah nilai prioritasnya.

Catatan: ID Proses Anda kemungkinan akan berbeda dari yang ada dalam contoh. LANGKAH 6:

Untuk melihat perintah mana yang berjalan di terminal saat ini, ketik perintah berikut, ketikkan

perintah ini di shell: **jobs**, maka akan ditampilkan informasi seperti yang terlihat pada gambar 10.7



Gambar 10.7

LANGKAH 7:

Selanjutnya, mulai perintah ping lain di latar belakang dengan mengetik berikut ini: **ping localhost > /dev/null &**, maka akan ditampilkan informasi seperti yang terlihat pada gambar 10.8

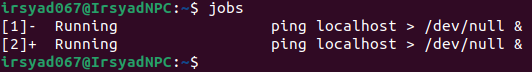
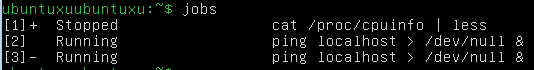


Gambar 10.8

Akan terlihat nomor pekerjaan dan ID proses yang berbeda untuk perintah baru ini.

LANGKAH 8:

Sekarang, harus ada dua perintah ping yang berjalan di latar belakang. Untuk memverifikasi, kasus ini berikan kembali perintah **jobs** lagi: **jobs**, maka akan ditampilkan informasi seperti yang terlihat pada gambar 10.10



Gambar 10.10

LANGKAH 9:

Setelah diverifikasi bahwa dua perintah ping sedang berjalan, bawa perintah pertama ke latar depan (foreground) dengan mengetik berikut ini: **fg %2**, maka akan ditampilkan informasi sepertiyang terlihat pada gambar 10.10



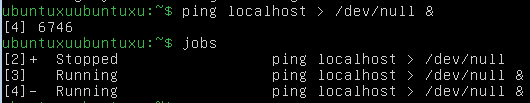
Gambar 10.10

Catatan: angka setelah tanda % adalah nilai dari job/pekerjaan, jadi tinggal sesuaikan saja angka job yang ada pada masing-masing komputer Anda.

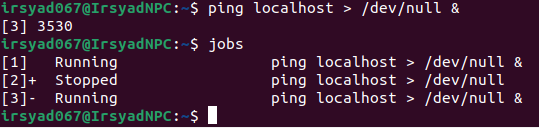
Untuk mengakhiri aktifitas efek dari perintah **fg %2**, ketikkan **ctrl+z**, agar aktifitas ping berhenti. Agar proses ini terus dijalankan di latar belakang, jalankan perintah berikut: **bg %2**.

LANGKAH 10:

Berikan kembali perintah ping di shell: **ping localhost > /dev/null &**, maka akan terlihat informasi jobs/pekerjaan dan PID, selanjutnya berikan perintah jobs di shell: **jobs** maka akan ditampilkan informasi seperti yang terlihat pada gambar 10.11.



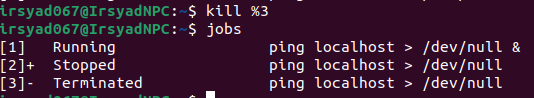
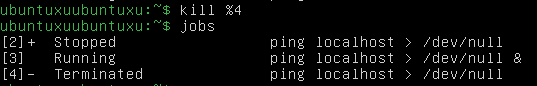




Gambar 10.11 Terlihat pada gambar ada 3 proses yang sedang berjalan. LANGKAH 11:

Menggunakan nomor pekerjaan, hentikan perintah ping terakhir dengan perintah **kill** dan

verifikasikan dengan perintah jobs apakah proses tersebut telah berhenti atau belum, dari kasus langkah 10, terlihat bahwa proses terakhir adalah dengan nomor pekerjaan [4], ketikkan perintah berikut di shell: **kill %4**, tekan enter, kemudian lanjutkan dengan perintah **jobs**, seperti terlihat pada gambar 10.12.

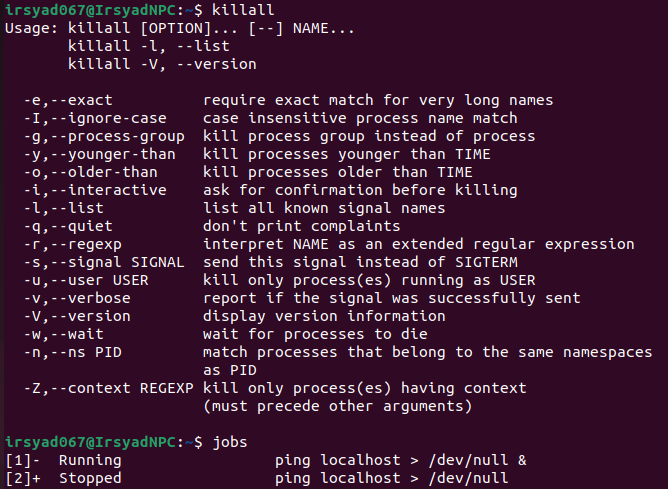


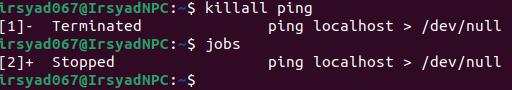
Gambar 10.12

Terlihat bahwa job nomor 4 telah berhenti, seperti yang ditunjukkan oleh gambar 10.12. LANGKAH 12:

Akhirnya, kita bisa menghentikan semua perintah ping dengan perintah **killall**. Setelah

menjalankan perintah **killall**, tunggu beberapa saat, dan kemudian jalankan perintah **jobs** untuk memverifikasi bahwa semua proses telah berhenti, berikan perintah **killall ping** di shell, dan kemudian lihat hasilnya dengan perintah **jobs**.

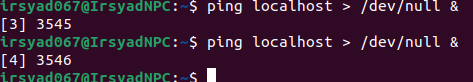




LANGKAH 13:

Kita akan menggunakan perintah **top** untuk bekerja dengan proses. Secara default program top mengurutkan proses dalam urutan persentase penggunaan CPU, sehingga program tersibuk akan berada di bagian atas daftar.

Berikan perintah: **ping localhost > /dev/null &**, sebanyak dua kali, seperti terlihat pada gambar 10.13.

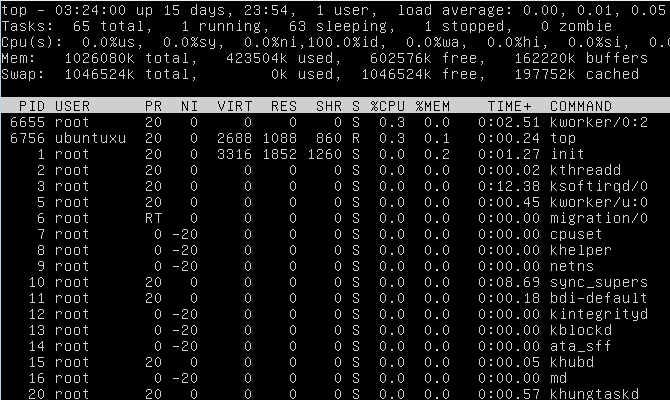


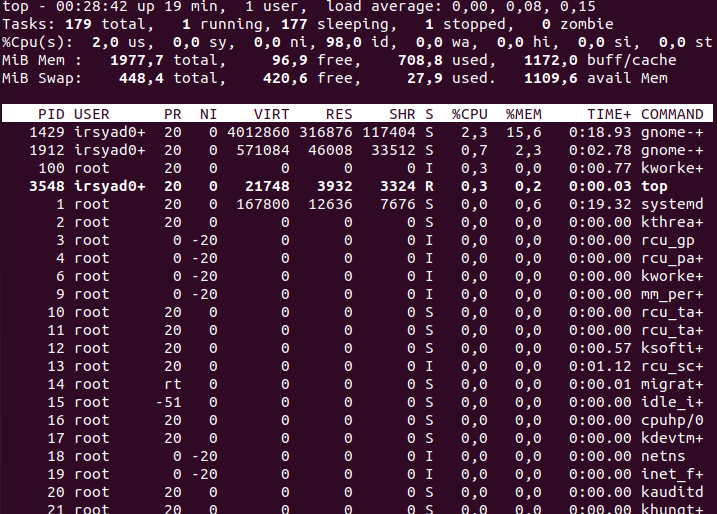
Gambar 10.13

Catat output PID dengan perintah di atas! Mereka akan berbeda dari contoh yang akan diberikan. Kita akan menggunakan PID pada langkah selanjutnya.

LANGKAH 14:

Berikan perintah: **top**, maka akan ditampilkan informasi seperti yang terlihat pada gambar 10.14.

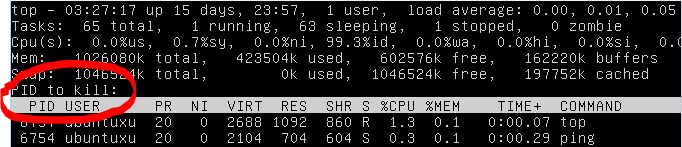


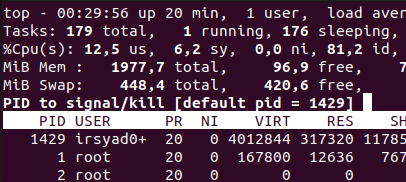


Gambar 10.14 Catatan: output dari perintah top akan berubah setiap 2 detik LANGKAH 15:

Perintah **top** adalah program interaktif, yang berarti kita dapat mengeluarkan perintah di dalam

program. Kita akan menggunakan perintah top untuk mengakhiri proses ping. Pertama-tama ketik huruf **k**. Perhatikan prompt telah muncul di bawah Swap:



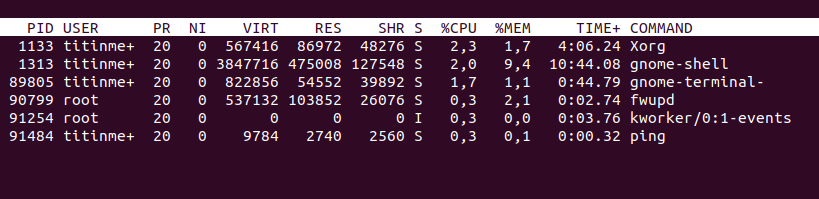
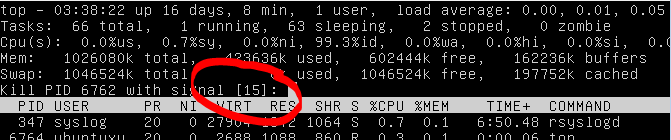


Gambar 10.15

LANGKAH 16:

Di PID untuk mengakhiri: ketikkan PID dari proses ping yang berjalan pertama, dari gambar

10.13 terlihat salah satu PID ping adalah **6754**, jadi ketikkan nilai tersebut di bagian kill yang ditandai dengan lingkaran merah pada gambar 10.15, setelah itu tekan enter, maka akan tampil seperti pada gambar 10.16.



Gambar 10.16

Pada Kill PID dengan sinyal [15]: ... masukkan sinyal untuk mengirim ke proses ini. Dalam hal ini, cukup tekan tombol Enter untuk menggunakan sinyal default. Perhatikan bahwa perintah ping pertama dihapus dan hanya satu perintah ping yang tersisa dalam daftar (Anda mungkin perlu menunggu beberapa detik saat perintah teratas di-refresh).

Ada beberapa nilai numerik berbeda yang dapat dikirim ke suatu proses. Ini adalah nilai yang telah ditentukan, masing-masing dengan makna yang berbeda. Jika Anda ingin mempelajari lebihlanjut tentang nilai-nilai ini, ketikkan man kill di terminal.

Prompt menunjukkan bahwa sinyal default adalah sinyal terminasi yang ditunjukkan oleh SIGTERM atau angka 15.

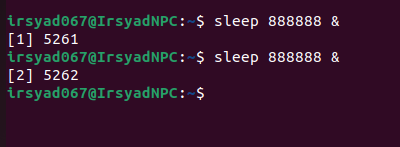
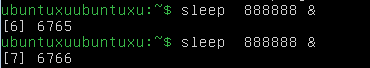
Selain angka default 15 yang ada, kita juga dapat memberikan nilai/angka 9 untuk melakukan terminasi sebuah proses, karena nilai 9 adalah signal untuk kill proses dan untuk keluar dari halaman top kita dapat melakukannya dengan menekan tombol q.

LANGKAH 17:

Dalam langkah ini, kita akan terus bekerja dengan proses. kita akan menggunakan **pkill** dan **kill**

untuk menghentikan proses.

ketikkan perintah berikut di shell: **sleep 888888 &**, sebanyak dua kali, seperti terlihat pada gambar 10.17.

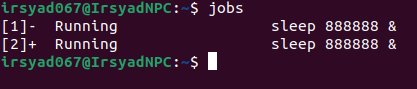
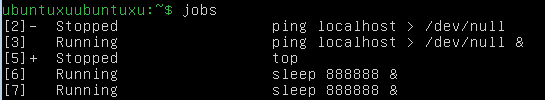


Gambar 10.17

Perintah sleep biasanya digunakan untuk menjeda program (skrip shell) untuk periode waktu tertentu. Dalam hal ini digunakan hanya untuk memberikan perintah yang akan memakan waktu lama untuk dijalankan.

LANGKAH 18:

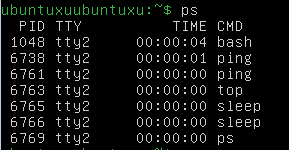
Jalankan perintah: **jobs**, dan lihat hasilnya.

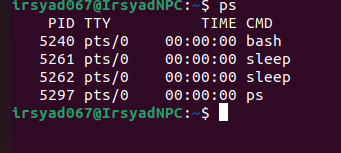


Gambar 10.18

LANGKAH 19:

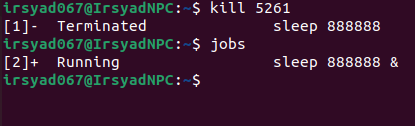
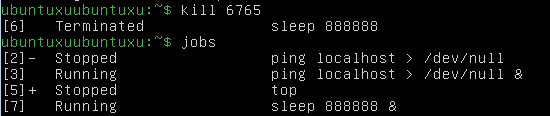
Sekarang, gunakan perintah kill untuk menghentikan instance pertama dari perintah sleep dengan mengetik berikut ini (gantilah PID dengan id proses dari perintah sleep pertama Anda). Juga, jalankan perintah **jobs** untuk memverifikasi bahwa proses telah dihentikan, untuk melihat PID/id proses bisa menggunakan perintah **ps**, seperti terlihat pada gambar 10.19.





Gambar 10.19

Lalu berikan perintah: **kill 6765**, lalu lihat dengan perintah **jobs**, maka akan terilhat proses sleep dengan PID 6765 telah hilang, seperti terlihat pada gambar 10.20.



Gambar 10.20

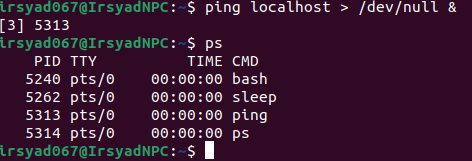
Jika kita ingin menghentikan proses dengan pkill dari perintah sleep yang tersisa, maka harus menggunakan nama program dan bukan PID, cukup ketikkan perintah: **pkill -15 sleep**. 15 adalah opsi sinyal default untuk menghentikan proses.

LANGKAH 20:

Perintah **ps** dapat digunakan untuk melihat proses. Secara default, perintah **ps** hanya akan menampilkan proses yang berjalan di shell saat ini.

Mulai proses latar belakang (background) dengan memberikan perintah: **ping localhost >**

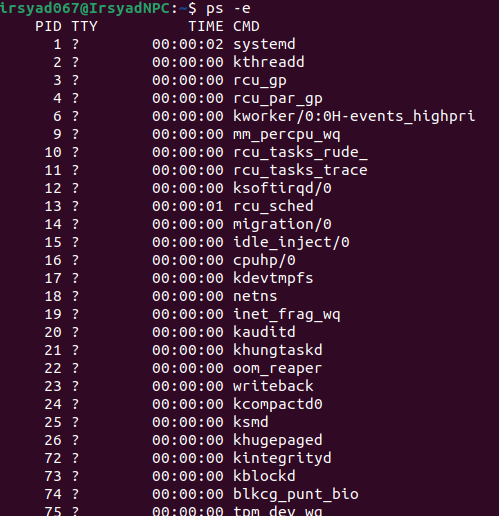
**/dev/null &**, dan lihat proses saat ini menggunakan perintah ps, seperti terlihat pada gambar 10.21.

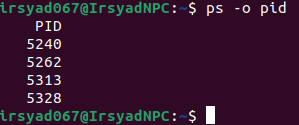


Gambar 10.21 Terlihat ada proses ping dengan PID: 6781.

LANGKAH 21:

Menampilkan semua proses yang ada dapat dilakukan dengan memberikan perintah: **ps –e**, karena lingkungan ini merupakan sistem operasi tervirtualisasi, ada jauh lebih sedikit proses daripada apa yang biasanya ditunjukkan dengan Linux yang berjalan langsung pada perangkat keras. Jika ingin menampilkan berdasarkan kolom dapat menggunakan opsi –o kemudian diikuti dengan nama kolom yang ingin ditampilkannya, misal berikut ini: **ps -o pid**.





LANGKAH 22:

Log sistem sangat penting untuk banyak tugas, termasuk memperbaiki masalah sistem operasi dan memastikan bahwa sistem kita aman. Mengetahui di mana file log sistem disimpan dan bagaimana cara memeliharanya penting bagi administrator sistem.

Ada dua daemon yang menangani pesan log: **daemon syslogd** dan **daemon klogd**. Biasanya Anda tidak perlu khawatir tentang klogd; ia hanya menangani pesan log kernel dan mengirimkan informasi lognya ke daemon syslogd.

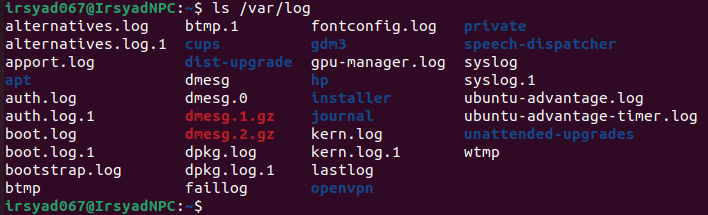
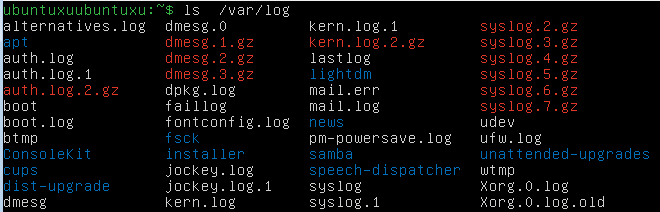
Pesan yang dihasilkan oleh kernel pada saat boot disimpan dalam file **/var/log/dmesg**. Perintah **dmesg** memungkinkan tampilan pesan kernel saat ini, serta memberikan kontrol apakah pesan itu akan muncul di jendela konsol terminal.

File log utama yang ditulis oleh syslogd adalah **/var/log/messages**.

Selain logging yang dilakukan oleh **syslogd**, banyak proses lain melakukan logging sendiri. Beberapa contoh proses yang melakukan logging sendiri termasuk server web Apache (file log berada di direktori /var/log/httpd), kebanyakan sistem unix mencatatnya di sistem (/var/log/cups) dan daemon auditd (/var/log/audit).

Karena beberapa perintah berikutnya yang akan dijalankan memerlukan hak pengguna super user(root), gunakan perintah su untuk beralih ke akun root. Jadi pastikan Anda sebagai user root!.

Log sistem disimpan di direktori /var/log. Daftar file dalam direktori ini dapat dilihat dengan memberikan perintah: **ls /var/log**, maka akan ditampilkan informasi seperti yang terlihat pada gambar 10.22.



Gambar 10.22

LANGKAH 23:

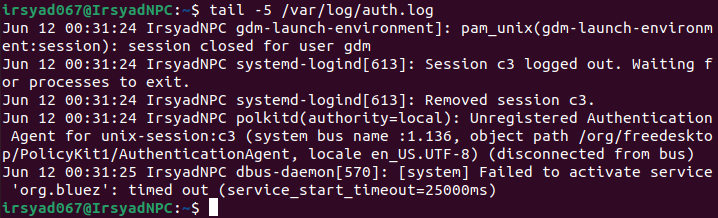
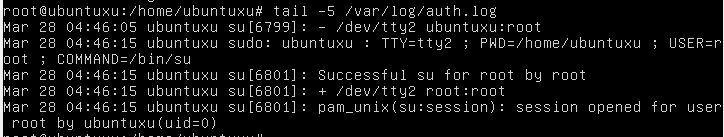
Setiap file log mewakili layanan atau fitur. Misalnya, file **auth.log** menampilkan informasi mengenai otorisasi atau otentikasi, seperti upaya login pengguna. Data baru disimpan di bagian bawah file. Jalankan perintah berikut untuk melihat contoh file log: **ssh localhost**, maka akan ditampilkan

informasi seperti yang terlihat pada gambar 10.23.



Gambar 10.23

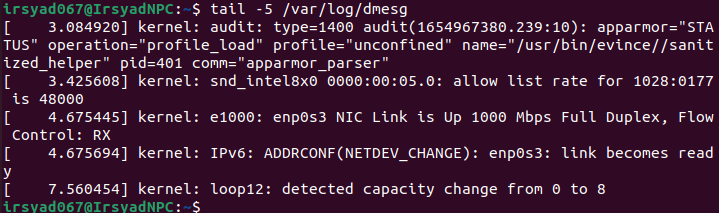
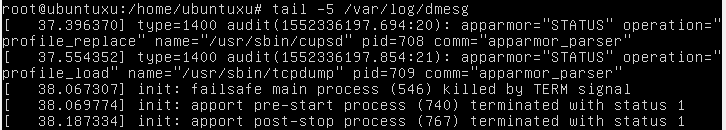
Perintah **ssh** digunakan untuk menghasilkan data dalam file **/var/log/auth.log**. Perhatikan bahwa upaya login yang gagal dicatat dalam file /var/log/auth.log, log dari akses yang gagal dapat dilihat menggunakan perintah: **tail -5 /var/log/auth.log**, dan outputnya seperti terlihat pada gambar 10.24



Gambar 10.24

LANGKAH 24:

Lihat lima baris terakhir dari file **/var/ log/dmesg** untuk melihat pesan kernel dari waktu boot dan jalankan perintah dmesg yang disalurkan ke perintah tail untuk melihat lima pesan kernel terakhir. Ketikkan perintah berikut di shell: **tail -5 /var/log/dmesg**, dan outputnya seperti terlihat pada gambar 10.25.



Gambar 10.25

Saat ini kita mungkin berpikir "apa arti semua pesan ini?". Jawaban untuk pertanyaan itu bukanlah

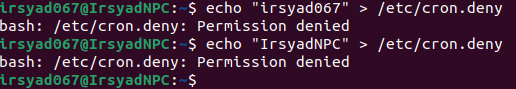
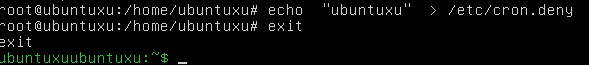
jawaban yang sederhana, namun poin di balik pelajaran ini bukan untuk menjelaskan arti dari semua pesan log, melainkan untuk mempelajari di mana menemukan pesan log.

Ketika kita mendapatkan lebih banyak pengalaman di Linux, kita akan mulai memecahkan masalah. Dalam kebanyakan kasus, tempat pertama yang ingin dilihat adalah file log.

Untuk memberikan contoh pemecahan masalah yang realistis, ikuti serangkaian tugas berikutnya.

LANGKAH 25:

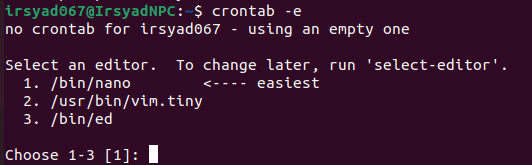
Masukkan perintah berikut untuk menonaktifkan kemampuan pengguna ubuntuxu dari membuat entri crontab dan keluar lalu kembali masuk sebagai pengguna ubuntuxu. Tentu saja ini disesuaikan dengan nama pengguna yang ada di sistem Anda, bisa saja nama user di sistem Anda adalah student, maka ganti ubuntuxu dengan student. Jalankan perintah seperti yang terlihat pada gambar 10.26, yaitu: **echo “ubuntuxu” > /etc/cron.deny**, kemudian **exit**, maka sistem akan kembalike shell pengguna.



Gambar 10.26

LANGKAH 26:

Coba jalankan perintah crontab berikut: **crontab –e**, maka outputnya seperti terlihat pada gambar 10.27.



Gambar 10.27

Pada gambar terlihat bahwa user tersebut tidak diizinkan untuk membuat program crontab.

Catatan: Perintah ini gagal karena entri di **/etc/cron.deny**. Jika nama pengguna ada di file ini, maka pengguna itu tidak dapat menggunakan perintah crontab.

LANGKAH 25:

Daripada pengguna berada di mode root dengan perintah su, gunakan sudo untuk mengeksekusi perintah berikut dengan hak akses root:

LANGKAH 25:

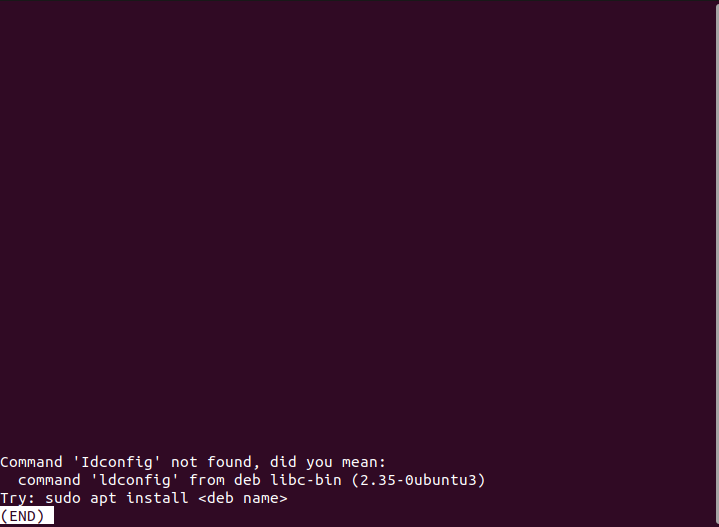
Shared libraries (Pustaka bersama) adalah file yang berisi kode yang dapat ditautkan oleh file program yang dapat dieksekusi untuk menggunakan kode itu. Karena beberapa program sering menautkan ke file librari tunggal, ini membantu mengurangi jumlah ruang yang diperlukan untuk kode ini karena setiap program tidak harus memiliki salinan kode librari sendiri.

File librari ini paling sering disimpan di direktori **/lib** dan **/usr/lib**. Direktori tambahan dapat ditambahkan dengan mengedit file konfigurasi, **/etc/ld.so.conf**. Anda juga dapat membuat file dengan nama yang berakhiran **.conf** dan menempatkannya di direktori **/etc/ld.so.conf.d**. Terakhir, Anda juga dapat mengatur variabel lingkungan LD\_LIBRARY\_PATH juga.

Ketika eksekusi sebagai pengguna root, perintah **ldconfig** dapat digunakan untuk memperbarui ***cache*** dan tautan simbolik untuk shared library (pustaka bersama) di sistem. Sebagai pengguna biasa, Anda akan menjalankan perintah **ldconfig** untuk mencetak daftar shared library ini.

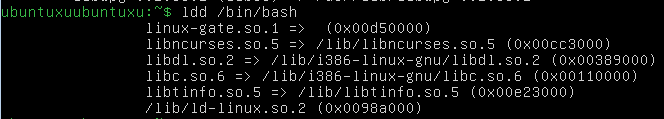
Jalankan perintah berikut di shell: **ldconfig -p | less**, dan jika ingin keluar dari halaman ini tekan

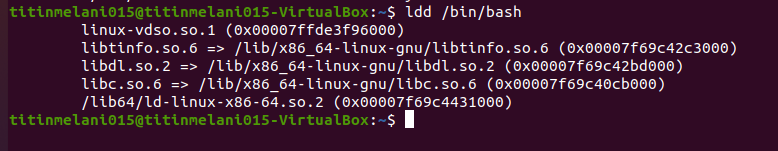
# tombol q

****

LANGKAH 25:

Untuk melihat librari apa yang ditautkan ke executable, seperti **/bin/bash**, jalankan perintah berikut di shell: **ldd /bin/bash**.





Gambar 10.28

KESIMPULAN:

Berikan kesimpulan Anda dari praktikum ini!

***Mengelola kemampuan untuk memuat librari, memeriksa file sistem pada direktori/proc,dan penggunaan perintah ps untuk melihat informasi proses pada sistem.***

SARAN:

Berikan saran jika ada!