2 Maret 2025

**LAPORAN**

Mata kuliah Workshop Administrasi Jaringan



Nama dosen pengampu:

Pak Dr Ferry Astika Saputra ST, M.Sc

Dikerjakan oleh

Nama : Yofi Yudistian

NRP : 3123600010

Kelas :  2 D4 IT A

**Chapter 4: Proses control**

**Proses control** terdiri dari sebuah address space dan sekumpulan struktur data didalam kernel. Address space adalah sekumpulan halaman memori yang ditandai oleh kernel untuk proses penggunaan. (Halaman adalah unit dimana memori di kelola, biasanya berukuran 4Kib atau 8Kib). Halaman digunakan untuk menyimpan kode, data, tumpukan proses.

Struktur data kernrel mencatat berbagai informasi tentang:

* Peta ruang alamat proses
* Status proses saat ini(berjalan, tidur, dst)
* Informasi tentang sumber daya yang digunakan proses(cpu, memoei, dst)
* Informasi tentang file dan port jaringan yang telah dibuka oleh proses
* Sinyal proses yang ditutup(sinyal yang sekarang diblokir)
* Pemilik proses(id pengguna yang memulai proses)

**Thread** adalah konteks eksekusi dalam sebuah proses. Sebuah proses dapat memiliki beberapa thread, yang semuanya berbagi address space dan sumber daya lain yang sama.thread digunakan untuk mencapai paralelisme dalam proses. Thread juga dikenal sebagai proses yang ringan karena jauh lebih murah untuk dibuat dan menghancurkan proses.

Contohnya untuk memahami konsep proses dan thread, pertimbangkan server web. Server web mendengarkan koneksi yang masuk dan kemudian membuat thread baru untuk menangani setiap perminataan yang masuk. Setiap thread menangani satu perminataan dalam satu waktu, tapi server web secara keseluruhan bisa menangani banyak permintaan secara bersamaan karena memiliki banyak thread. Disini server web adalah proses dan setiap thread adalah konteks eksekusi yang terpisah didalam proses.

**PID: Proses ID**

Setiap proses diidentifikasi dengan nomor id yang unik atau pid. Pid adalah bilangan bulat yang diberikan kernel kepada setiap proses yang dibuat. Pid digunakan untuk merujuk ke proses dalam berbagai panggilan sistem, misalnya untuk mengirim sinyal ke proses.

Konsep namespace memungkinkan proses yang berbeda memiliki pid yang sama. Namespace digunakan untuk membuiat container, yang merupakan lingkungan terisolasi yang memiliki pandangan mereka sendiri terhadap sistem. Kontainer digunakan untuk menjalankan beberapa contoh aplikasi pada sistem yang sama, masing-masing dalam lingkungan terisolasi.

**PPID: parent process ID number**

Setiap proses dikaitkan dengan proses induk, yaitu proses yang membuatnya. Proses induk id,atau ppid adalah pid dari proses induk. Ppid dignakan untuk merujuk ke proses induk dalam berbagai panggilan sistem, misalnya, untuk mengirim sinyal ke proses induk.

**UID & EUID: id user dan id effective user**

Id pengguna atau uid adalah id pengguna yang memulai proses. Id pengguna efektif atau euid adalah id pengguna yang digunakan untuk menentukan sumber daya apa yang dapat diakses oleh proses. Euid digunakan untuk mrngontrol akses ke file, port jaringan dan sumber daya lain.

**Lifecycle of a process**

Untuk membuat proses baru, sebuah proses menyalin dirinya sendiri dengan panggilan sistem fork. fork membuat salinan dari proses asli, dan salinan itu sebagian besar identik dengan induknya. Proses baru ini memiliki PID yang berbeda dan memiliki informasi akuntansinya sendiri. (Secara teknis, sistem Linux menggunakan clone, sebuah superset dari fork yang menangani thread dan menyertakan fitur-fitur tambahan. fork tetap berada di dalam kernel untuk kompatibilitas ke belakang, tetapi memanggil clone secara internal).

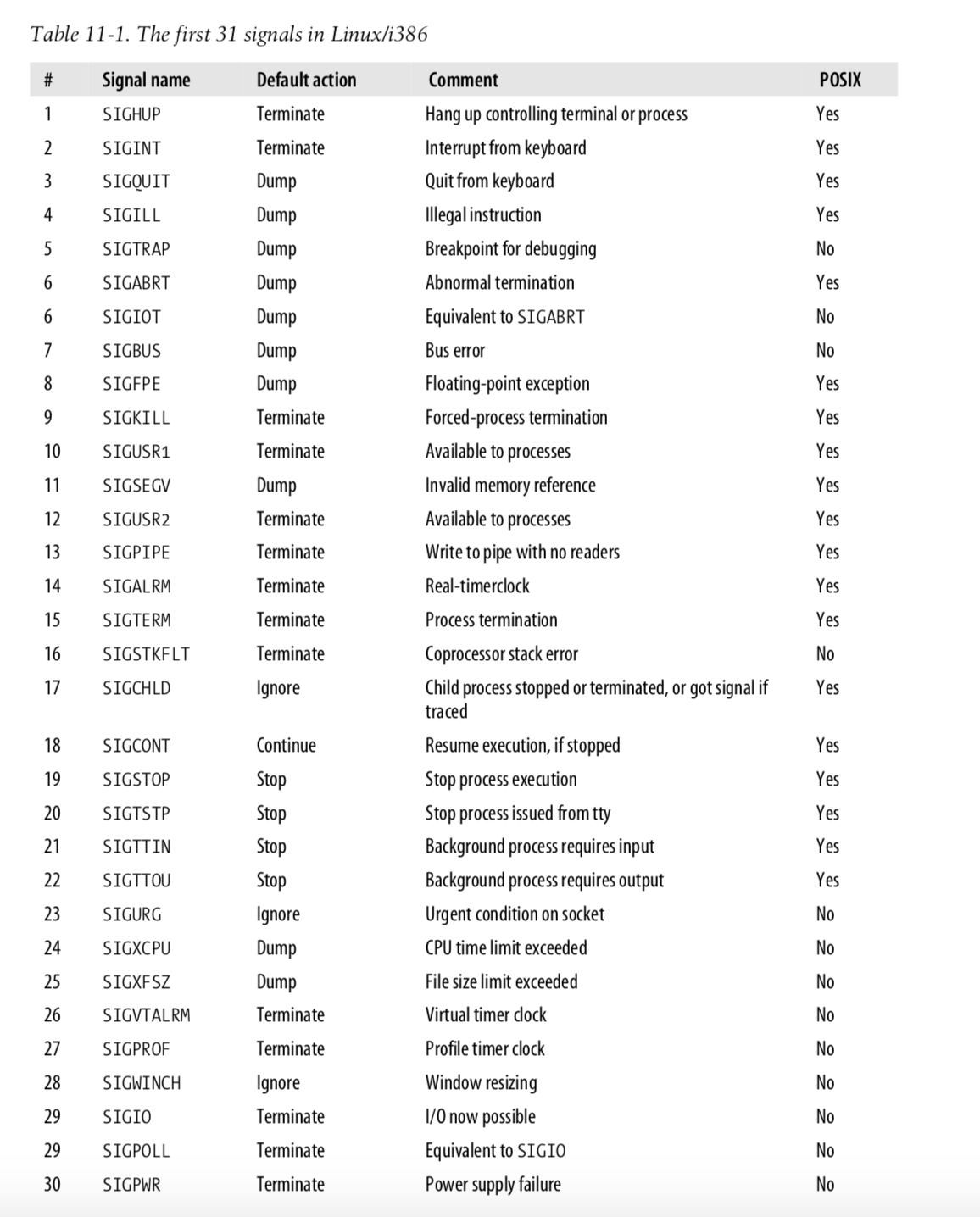
Ketika sistem melakukan booting, kernel secara otonom membuat dan menginstall beberapa proses. Yang paling terkenal adalah init atau systemd, yang selalu menjadi proses nomor 1. Proses ini mengeksekusi skrip startup sistem, walaupun cara yang tepat untuk melakukan hal ini sedikit berbeda antara UNIX dan Linux. Semua proses selain yang dibuat oleh kernel adalah turunan dari proses primordial ini.

**signal**

Sinyal adalah cara untuk mengirim pemberitahuan ke suatu proses. Sinyal digunakan untuk memberi tahu suatu proses bahwa peristiwa tertentu telah terjadi.

Sekitar tiga puluh jenis yang berbeda didefinisikan, dan mereka digunakan dalam berbagai cara:

* Mereka dapat dikirim di antara proses sebagai sarana komunikasi.
* Mereka dapat dikirim oleh driver terminal untuk membunuh, menginterupsi, atau menangguhkan proses ketika tombol seperti dan ditekan.
* Mereka dapat dikirim oleh administrator (dengan kill) untuk mencapai berbagai tujuan.
* Mereka dapat dikirim oleh kernel ketika sebuah proses melakukan pelanggaran seperti pembagian dengan nol.
* Mereka dapat dikirim oleh kernel untuk memberitahukan sebuah proses tentang sebuah kondisi yang “menarik” seperti matinya sebuah proses anak atau tersedianya data pada sebuah saluran I/O.



Sinyal KILL, INT, TERM, HUP, dan QUIT terdengar seolah-olah memiliki arti yang kurang lebih sama, tetapi penggunaannya sebenarnya sangat berbeda.

* KILL tidak dapat diblokir dan menghentikan proses pada tingkat kernel. Sebuah proses tidak akan pernah benar-benar menerima atau menangani sinyal ini.
* INT dikirim oleh driver terminal ketika pengguna mengetikkan . Ini adalah permintaan untuk menghentikan operasi yang sedang berjalan. Program sederhana harus berhenti (jika mereka menangkap sinyal tersebut) atau membiarkan diri mereka dibunuh, yang merupakan kondisi default jika sinyal tersebut tidak tertangkap. Program yang memiliki baris perintah interaktif (seperti shell) harus menghentikan apa yang mereka lakukan, membersihkan, dan menunggu masukan dari pengguna lagi.
* TERM adalah permintaan untuk menghentikan eksekusi sepenuhnya. Diharapkan proses yang menerima akan membersihkan statusnya dan keluar.
* HUP dikirim ke sebuah proses ketika terminal pengendali ditutup. Awalnya digunakan untuk mengindikasikan “menutup” koneksi telepon, sekarang sering digunakan untuk menginstruksikan proses daemon untuk mengakhiri dan memulai ulang, sering kali untuk mempertimbangkan konfigurasi baru. Perilaku yang tepat tergantung pada proses spesifik yang menerima sinyal HUP.
* QUIT mirip dengan TERM, kecuali bahwa ia secara default menghasilkan core dump jika tidak tertangkap. Beberapa program mengkanibalisasi sinyal ini dan menafsirkannya sebagai sesuatu yang lain.

**kill: sned signal**

Sesuai dengan namanya, perintah kill paling sering digunakan untuk menghentikan sebuah proses. kill dapat mengirimkan sinyal apa saja, tetapi secara default ia mengirimkan TERM. kill dapat digunakan oleh pengguna normal pada proses mereka sendiri atau oleh root pada proses apa pun. Sintaksnya adalah:

kill [-signal] pid

Dimana signal adalah nomor atau nama simbolik dari sinyal yang akan dikirim dan pid adalah nomor identifikasi proses dari proses target. Sebuah kill tanpa nomor sinyal tidak menjamin bahwa proses akan mati, karena sinyal TERM dapat ditangkap, diblokir, atau diabaikan. Perintah kill -9 pid dijamin akan membunuh proses karena sinyal KILL tidak dapat ditangkap, diblokir, atau diabaikan.

killall membunuh proses berdasarkan nama, bukan berdasarkan ID proses. Perintah ini tidak tersedia pada semua sistem. Contoh:

killall firefox

Perintah pkill mirip dengan killall tetapi menyediakan lebih banyak pilihan.

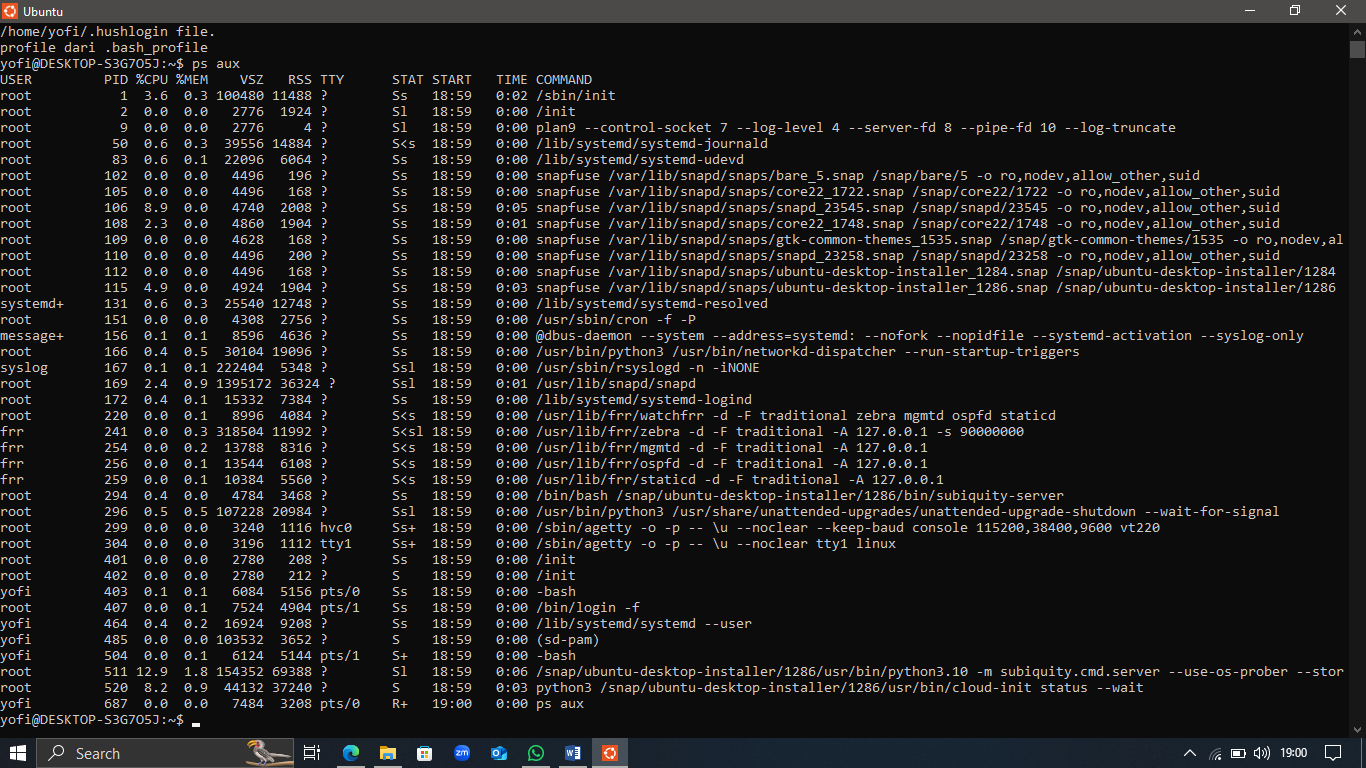
pkill -u abdoufermat # kill all processes owned by user abdoufermat

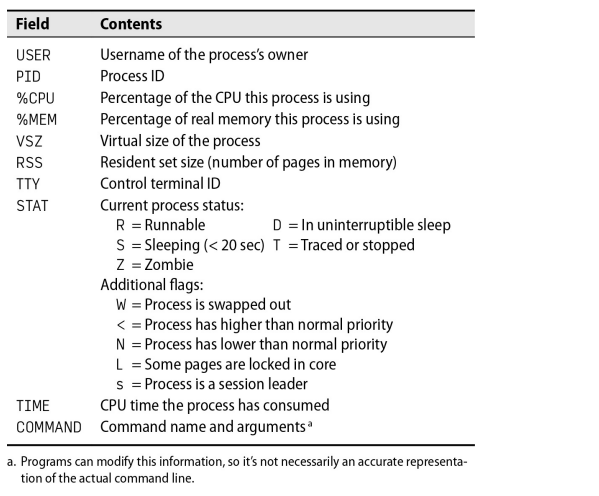
**PS: monitoring processes**

Perintah ps adalah alat utama administrator sistem untuk memantau proses. Meskipun versi ps berbeda dalam argumen dan tampilannya, semuanya memberikan informasi yang pada dasarnya sama.

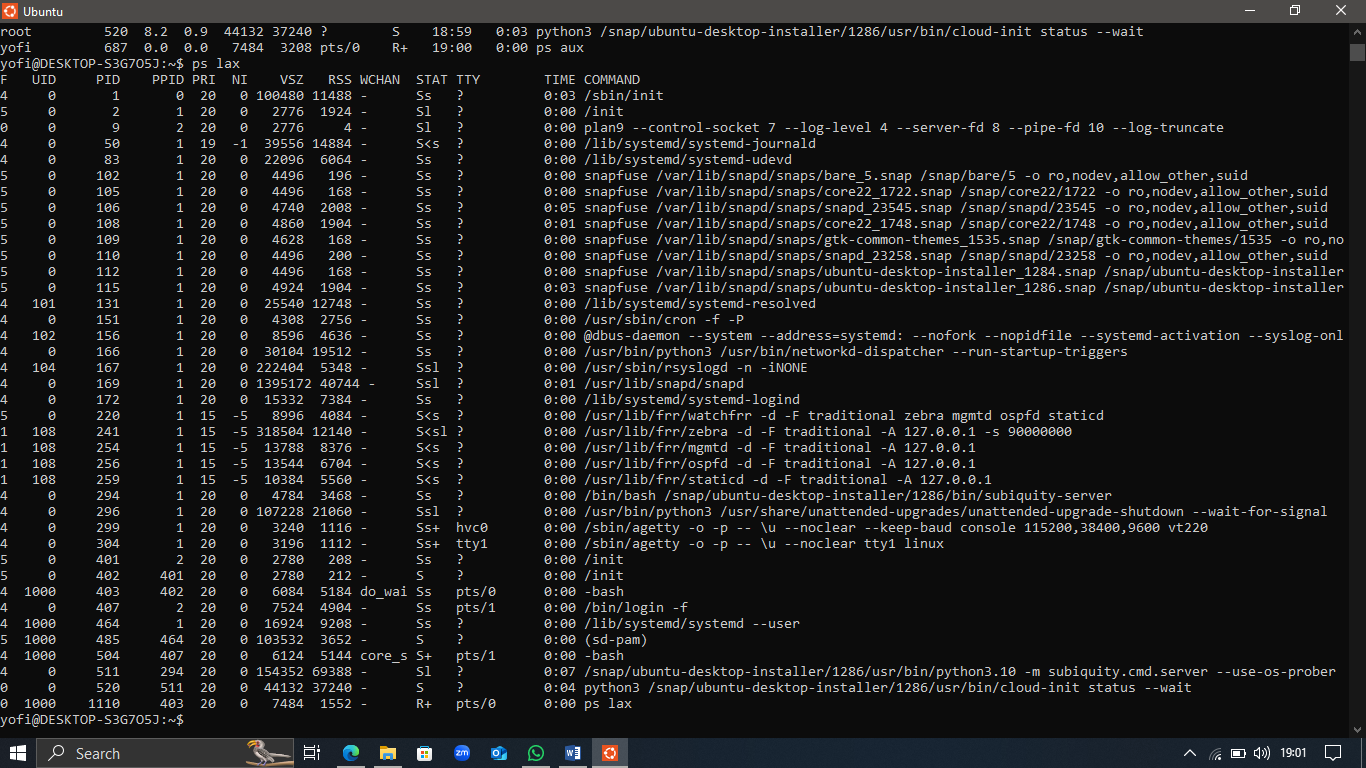
ps dapat menampilkan PID, UID, prioritas, dan terminal kontrol proses. Ia juga menginformasikan kepada Anda berapa banyak memori yang digunakan sebuah proses, berapa banyak waktu CPU yang digunakan, dan apa statusnya saat ini (berjalan, berhenti, tidur, dan seterusnya).

Anda dapat memperoleh gambaran umum yang berguna dari sistem dengan menjalankan ps aux. Opsi a memberitahu ps untuk menampilkan proses dari semua pengguna, dan opsi u memberitahu ps untuk memberikan informasi rinci tentang setiap proses. Opsi x memerintahkan ps untuk menampilkan proses yang tidak berhubungan dengan terminal.



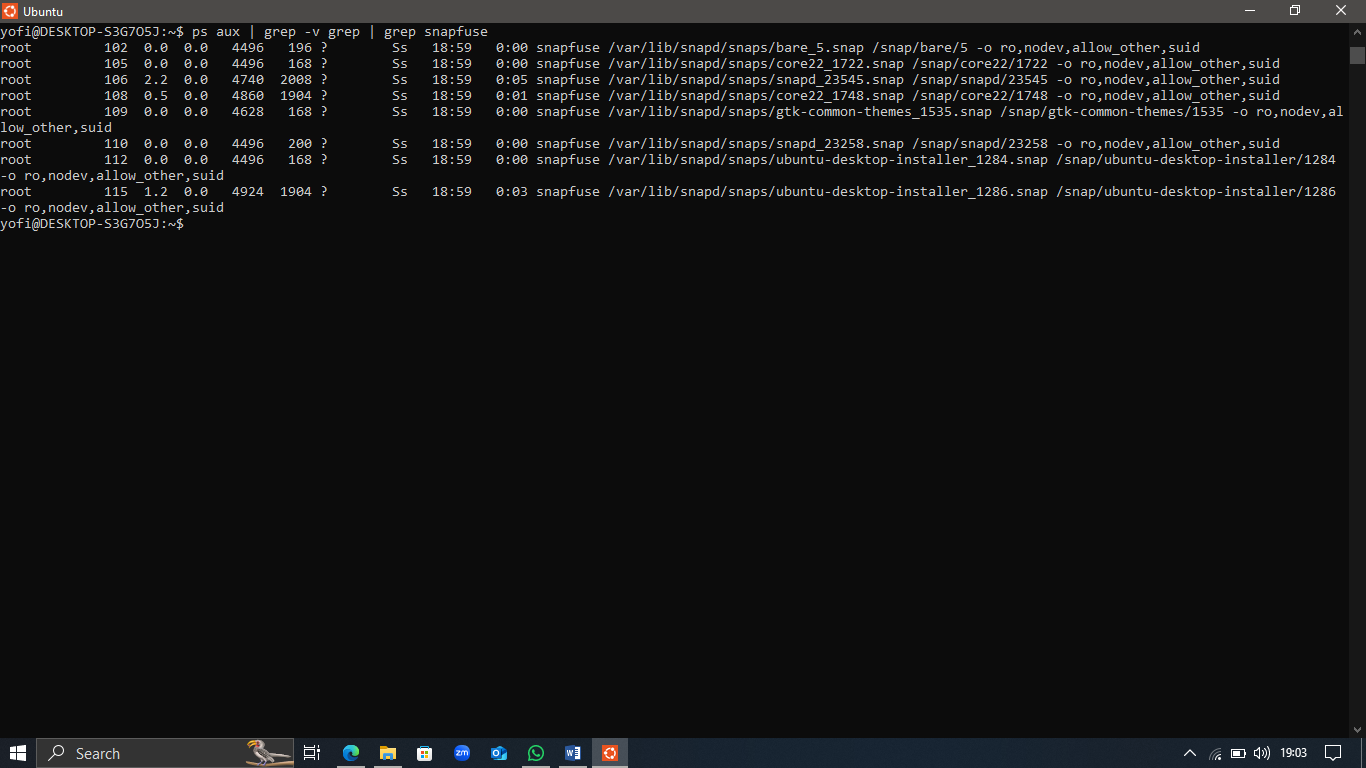


Satu set argumen lain yang berguna adalah lax, yang memberikan lebih banyak informasi teknis tentang proses. lax sedikit lebih cepat daripada aux karena tidak perlu menyelesaikan nama pengguna dan grup.



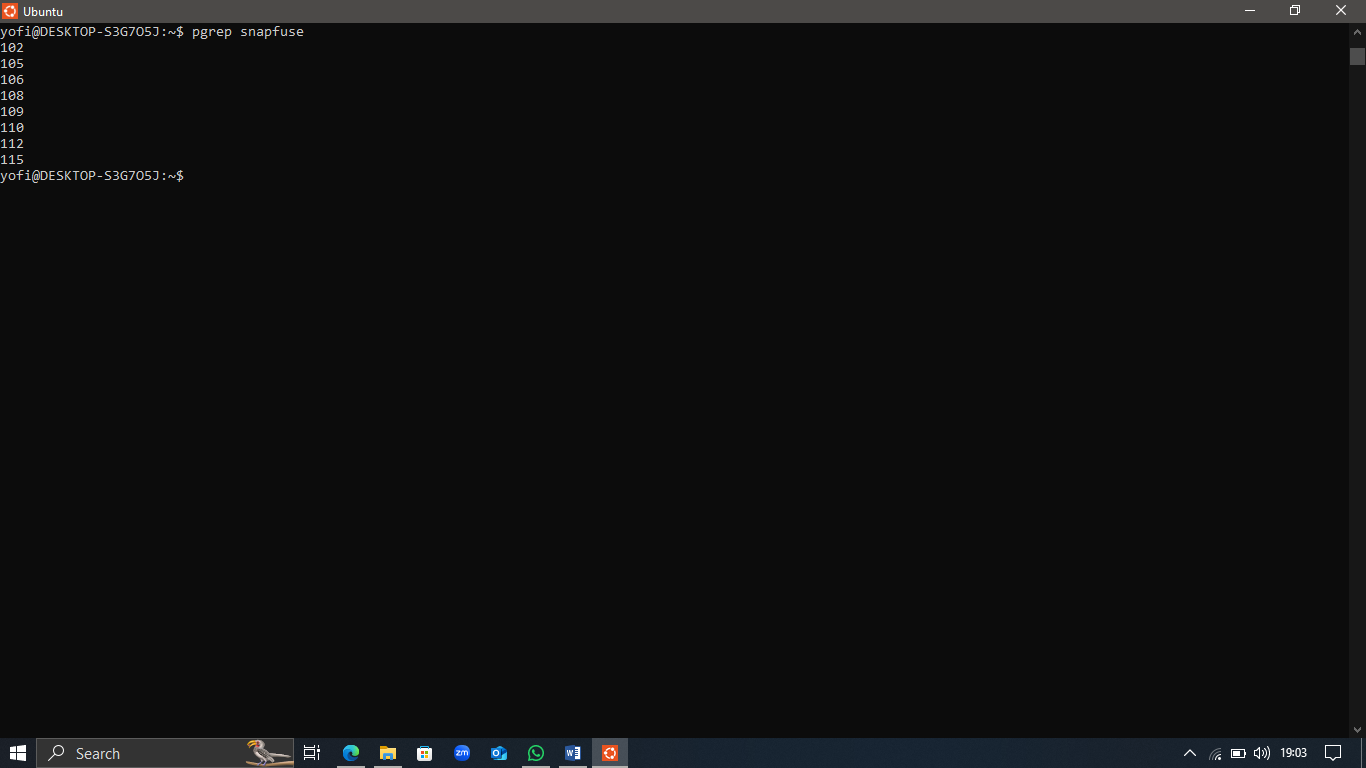
Untuk mencari proses tertentu, Anda dapat menggunakan grep untuk memfilter keluaran ps.

$ ps aux | grep -v grep | grep firefox



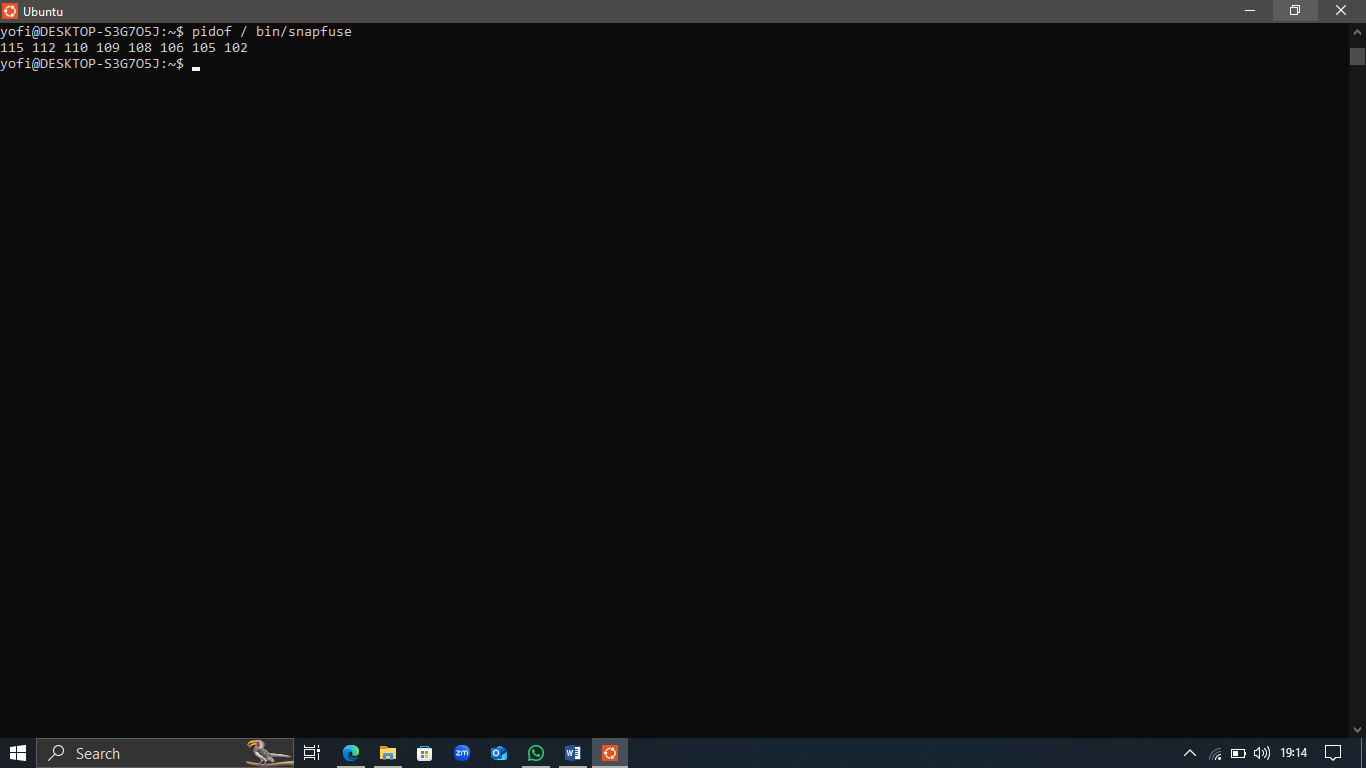
Kita dapat menentukan PID dari sebuah proses dengan menggunakan pgrep.

$ pgrep firefox



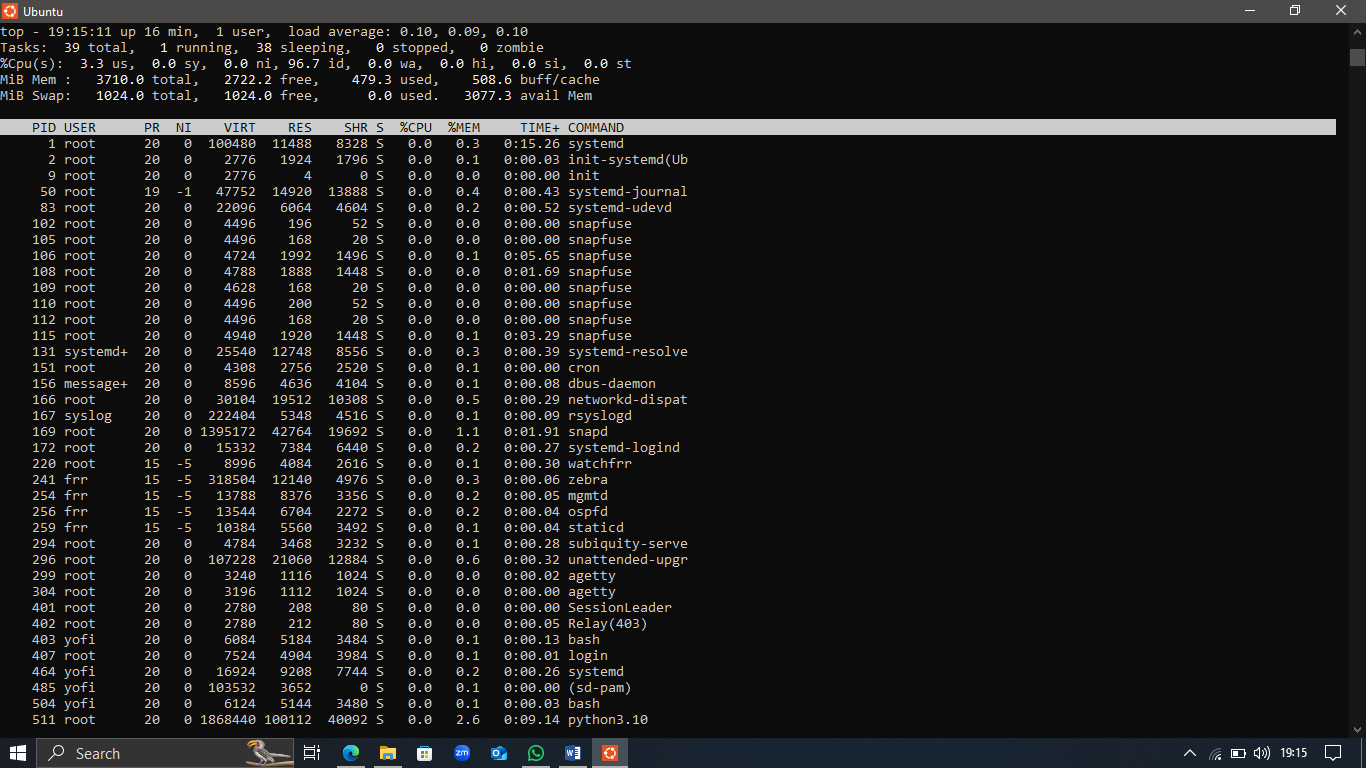
atau pidof.

$ pidof /usr/bin/firefox

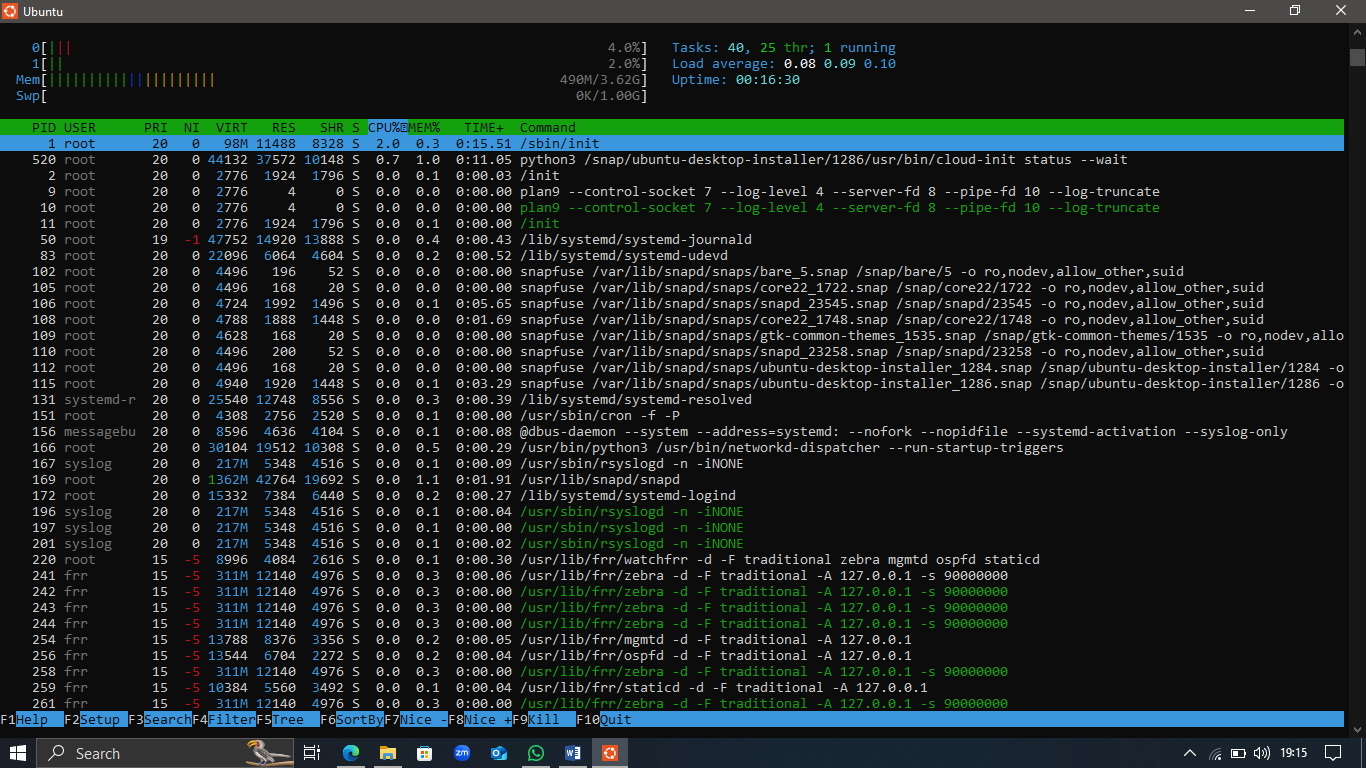


**Interactive monitoring with top**

Perintah top menyediakan tampilan real-time yang dinamis dari sistem yang sedang berjalan. Perintah ini dapat menampilkan informasi ringkasan sistem serta daftar proses atau thread yang saat ini dikelola oleh kernel Linux. Jenis informasi ringkasan sistem yang ditampilkan serta jenis, urutan, dan ukuran informasi yang ditampilkan untuk proses, semuanya dapat dikonfigurasi oleh pengguna dan konfigurasi tersebut dapat dipertahankan selama proses restart. Secara default, tampilan diperbarui setiap 1-2 detik, tergantung pada sistem.



Ada juga perintah htop, yang merupakan penampil proses interaktif untuk sistem Unix. Ini adalah aplikasi mode teks (untuk konsol atau terminal X) dan membutuhkan ncurses. Ini mirip dengan top, tetapi memungkinkan Anda untuk menggulir secara vertikal dan horizontal, sehingga Anda dapat melihat semua proses yang berjalan pada sistem, bersama dengan baris perintah lengkapnya. htop juga memiliki antarmuka pengguna yang lebih baik dan lebih banyak pilihan untuk operasi.



**Nice dan renice: change proses priority**

Nice adalah petunjuk numerik pada kernel tentang bagaimana proses harus diperlakukan dalam kaitannya dengan proses lain yang bersaing untuk mendapatkan CPU. Nilai niceness yang tinggi berarti prioritas yang rendah untuk proses Anda: Anda akan bersikap baik. Nilai yang rendah atau negatif berarti prioritas yang tinggi: Anda tidak terlalu baik! Kisaran nilai niceness yang diijinkan bervariasi di antara sistem. Pada Linux, kisarannya adalah -20 hingga +19, dan pada FreeBSD adalah -20 hingga +20.

Proses dengan prioritas rendah adalah proses yang tidak terlalu penting. Proses ini akan mendapatkan waktu CPU yang lebih sedikit daripada proses prioritas tinggi. Proses dengan prioritas tinggi adalah proses yang penting dan harus diberi waktu CPU lebih banyak daripada proses dengan prioritas rendah. Sebagai contoh, jika Anda menjalankan pekerjaan yang membutuhkan CPU intensif yang ingin Anda jalankan di latar belakang, Anda dapat memulainya dengan nilai niceness yang tinggi. Hal ini akan memungkinkan proses lain untuk berjalan tanpa diperlambat oleh pekerjaan Anda.

Perintah nice digunakan untuk memulai proses dengan nilai kebaikan tertentu. Sintaksnya adalah:

nice -n nice\_val [perintah]

# Contoh

nice -n 10 sh infinite.sh &

Perintah renice digunakan untuk mengubah nilai kebaikan dari proses yang sedang berjalan. Sintaksnya adalah:

renice -n nice\_val -p pid

# Contoh

renice -n 10 -p 1234

Nilai prioritas adalah prioritas aktual proses yang digunakan oleh kernel Linux untuk menjadwalkan sebuah tugas. Dalam sistem Linux, prioritas sistem adalah 0 hingga 139 di mana 0 hingga 99 untuk waktu nyata dan 100 hingga 139 untuk pengguna.

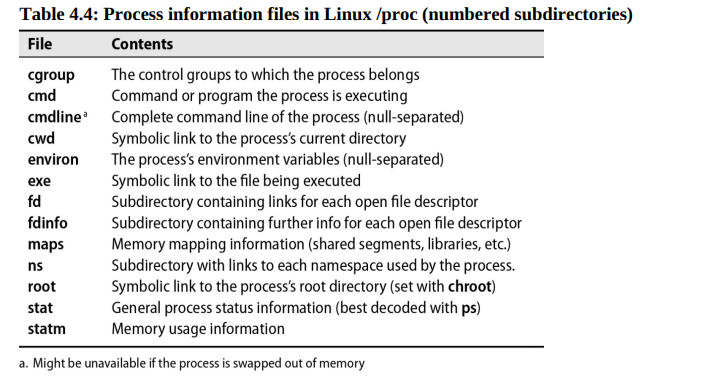
Hubungan antara nilai bagus dan prioritas adalah sebagai berikut:

nilai\_prioritas = 20 + nilai\_baik

Nilai default dari nice value adalah 0. Semakin rendah nilai nice value, semakin tinggi prioritas proses.

**The /proc filesystem**

Versi Linux ps dan top membaca informasi status proses dari direktori /proc, sebuah sistem berkas semu di mana kernel menampilkan berbagai informasi menarik tentang status sistem. Terlepas dari namanya, /proc berisi informasi lain selain proses (statistik yang dihasilkan oleh sistem, dll). Proses diwakili oleh direktori dalam /proc, dan setiap proses memiliki direktori yang diberi nama sesuai dengan PID-nya. Direktori /proc berisi berbagai macam berkas yang menyediakan informasi tentang proses, seperti baris perintah, variabel lingkungan, deskriptor berkas, dan sebagainya.



**Strace dan truss**

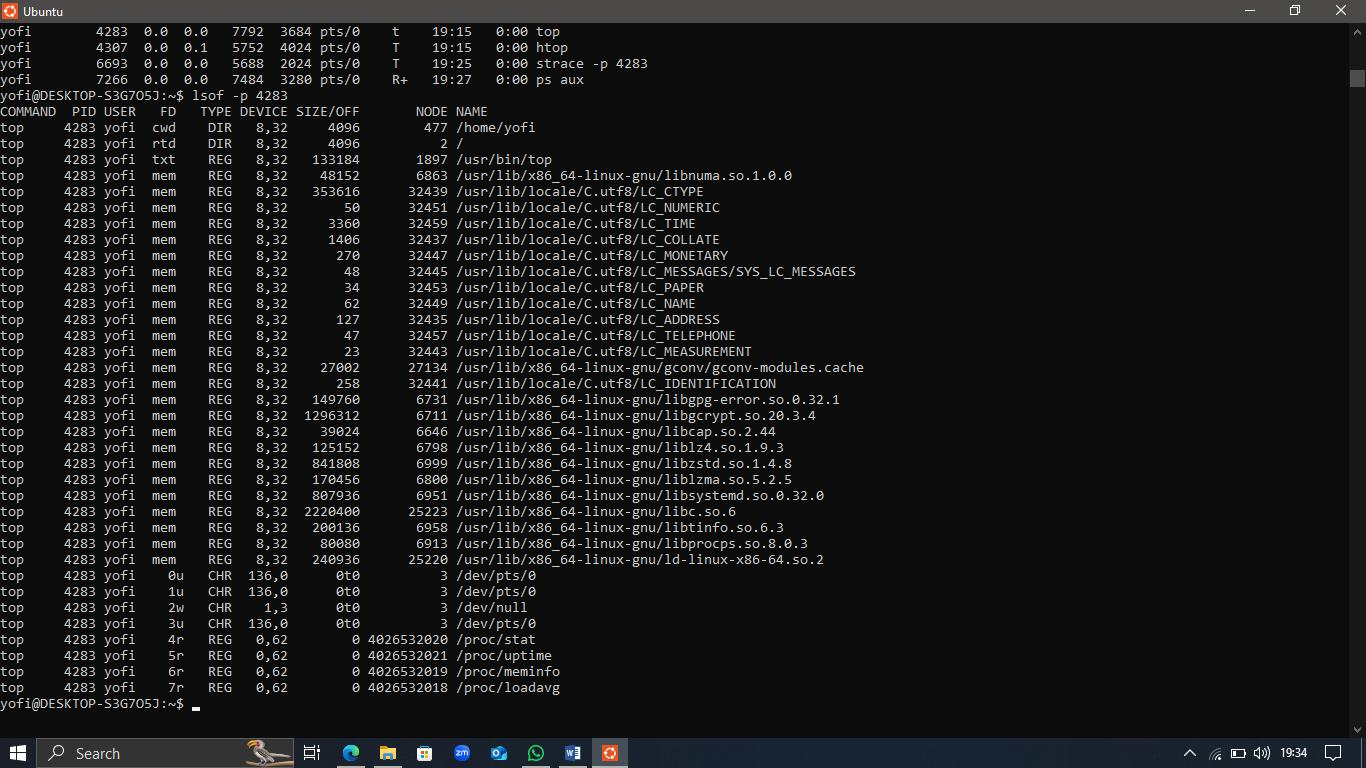
Untuk mengetahui apa yang sedang dilakukan oleh sebuah proses, Anda dapat menggunakan strace pada Linux atau truss pada FreeBSD. Perintah-perintah ini melacak panggilan sistem dan sinyal. Perintah-perintah ini dapat digunakan untuk men-debug sebuah program atau untuk memahami apa yang dilakukan oleh sebuah program. Sebagai contoh, log berikut ini dihasilkan oleh strace yang dijalankan pada sebuah salinan aktif dari top (yang berjalan sebagai PID 5810): strace -p 5810

**Runaway processes**

Kadang-kadang sebuah proses akan berhenti merespons sistem dan berjalan liar. Proses-proses ini mengabaikan prioritas penjadwalan mereka dan bersikeras untuk menggunakan 100% CPU. Karena proses lain hanya bisa mendapatkan akses terbatas ke CPU, mesin mulai berjalan sangat lambat. Ini disebut proses yang kabur. Perintah kill dapat digunakan untuk menghentikan proses yang kabur. Jika proses tidak merespon sinyal TERM, Anda dapat menggunakan sinyal KILL untuk menghentikannya.

Kita dapat menyelidiki penyebab proses runaway dengan menggunakan strace atau truss. Proses runaway yang menghasilkan output dapat memenuhi seluruh sistem berkas. Anda dapat menjalankan df -h untuk memeriksa penggunaan sistem berkas. Jika sistem berkas penuh, Anda dapat menggunakan perintah du untuk menemukan berkas dan direktori terbesar. Anda juga dapat menggunakan perintah lsof untuk mengetahui file mana yang dibuka oleh proses pelarian.

lsof -p pid



**Periodic processes**

**cron: schedule processes**

Cron adalah alat tradisional untuk menjalankan perintah pada jadwal yang telah ditentukan. Ia dimulai saat sistem melakukan booting dan berjalan selama sistem hidup. cron membaca berkas konfigurasi yang berisi daftar baris perintah dan waktu pemanggilannya. Baris perintah dieksekusi oleh sh, sehingga hampir semua hal yang dapat Anda lakukan secara manual dari shell juga dapat dilakukan dengan cron. Sebuah berkas konfigurasi cron disebut “crontab”, kependekan dari “cron table”. Crontab untuk pengguna individual disimpan di bawah /var/spool/cron (Linux) atau /var/cron/tabs (FreeBSD).

**format crontab**

File crontab memiliki lima bidang untuk menentukan hari, tanggal, dan waktu yang diikuti dengan perintah yang akan dijalankan pada interval tersebut.

**management crontab**

Perintah crontab digunakan untuk membuat, memodifikasi, dan menghapus crontab. Opsi -e digunakan untuk mengedit file crontab, opsi -l digunakan untuk membuat daftar file crontab, dan opsi -r digunakan untuk menghapus file crontab.

**Systemd timer**

Timer systemd adalah berkas konfigurasi unit yang namanya berakhiran .timer. timer systemd dapat digunakan sebagai alternatif untuk pekerjaan cron. Mereka lebih fleksibel dan lebih kuat daripada pekerjaan cron. Sebuah unit timer diaktifkan oleh unit layanan yang sesuai. Unit layanan dipicu oleh unit pengatur waktu pada waktu yang ditentukan dalam unit pengatur waktu. Unit pengatur waktu juga dapat diaktifkan oleh boot sistem atau oleh suatu peristiwa. Perintah systemctl digunakan untuk mengelola unit systemd. Opsi list-timers digunakan untuk membuat daftar timer yang aktif.

**Penggunaan umum untuk tugas terjadwal**

**Mengirim email**

Anda dapat secara otomatis mengirim email output laporan harian atau hasil eksekusi perintah menggunakan pengatur waktu cron atau systemd.

**Membersihkan sistem berkas**

Anda dapat menggunakan pengatur waktu cron atau systemd untuk menjalankan skrip yang membersihkan sistem berkas. Sebagai contoh, Anda dapat menggunakan skrip untuk membersihkan isi direktori sampah setiap hari pada tengah malam.

**Memutar file log**

Memutar file log berarti membaginya menjadi beberapa segmen berdasarkan ukuran atau tanggal, sehingga beberapa versi log yang lebih lama selalu tersedia. Karena rotasi log adalah peristiwa yang berulang dan terjadi secara teratur, maka ini adalah tugas yang ideal untuk dijadwalkan.

**Menjalankan pekerjaan batch**

Beberapa kalkulasi yang berjalan lama paling baik dijalankan sebagai pekerjaan batch. Sebagai contoh, pesan dapat terakumulasi dalam antrian atau database. Anda dapat menggunakan pekerjaan cron untuk memproses semua pesan yang mengantri sekaligus sebagai ETL (Ekstrak, Transformasi, Muat) ke lokasi lain, seperti gudang data.

**Mencadangkan dan mencerminkan**

Anda dapat menggunakan tugas terjadwal untuk mencadangkan direktori secara otomatis ke sistem jarak jauh. Mirror adalah salinan byte per byte dari sistem file atau direktori yang di-host di sistem lain. Mirror dapat digunakan sebagai bentuk pencadangan atau sebagai cara untuk mendistribusikan file di beberapa sistem. Anda dapat menggunakan eksekusi rsync secara periodik untuk menjaga mirror tetap mutakhir.

**Chapter 5: The Filesystem**

Tujuan dasar sistem berkas adalah untuk merepresentasikan dan mengatur sumber daya penyimpanan sistem. Sistem berkas dapat dianggap sebagai sebuah sistem yang terdiri dari empat komponen utama:

* Ruang nama - cara untuk menamai sesuatu dan mengaturnya dalam sebuah hirarki
* API - seperangkat panggilan sistem untuk menavigasi dan memanipulasi objek
* Model keamanan - skema untuk melindungi, menyembunyikan, dan berbagi sesuatu
* Implementasi - perangkat lunak untuk menghubungkan model logis ke perangkat keras

Sistem berkas berbasis disk yang utama adalah sistem berkas ext4, XFS, dan UFS, bersama dengan ZFS dan Btrfs dari Oracle. Masih banyak lagi yang lain yang tersedia, termasuk VxFS dari Verittas dan JFS dari IBM. Selain itu, kami juga memiliki beberapa sistem berkas asing seperti FAT dan NTFS yang digunakan oleh Windows dan sistem berkas ISO 9660 yang digunakan oleh CD dan DVD. Kebanyakan sistem berkas modern mencoba mengimplementasikan fungsionalitas sistem berkas tradisional dengan cara yang lebih cepat dan lebih dapat diandalkan, atau menambahkan fitur tambahan sebagai lapisan di atas semantik sistem berkas standar.

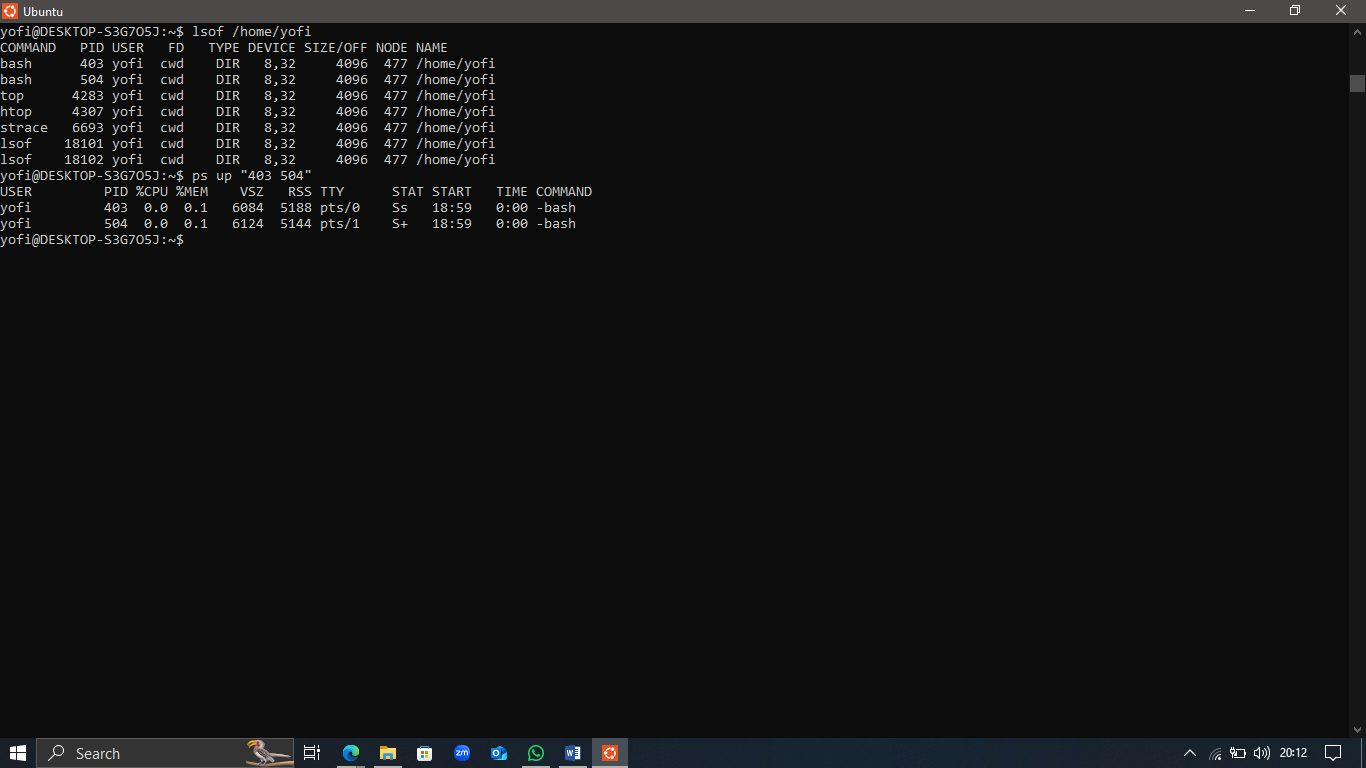
**Pathnames**

Kata “folder” hanyalah bocoran bahasa dari dunia Windows dan macOS. Artinya sama dengan direktori, yang lebih bersifat teknis. Daftar direktori yang mengarah ke sebuah berkas disebut nama path. Nama path adalah sebuah string yang menjelaskan lokasi file dalam hirarki sistem berkas. Nama path dapat berupa absolut (misalnya /home/username/file.txt) atau relatif (misalnya ./file.txt).

**Filesystem mouting & unmouting**

Sistem berkas terdiri dari potongan-potongan yang lebih kecil - disebut juga “sistem berkas” - yang masing-masing terdiri dari satu direktori dan subdirektori serta berkas-berkasnya. Kami menggunakan istilah pohon berkas untuk merujuk pada tata letak keseluruhan dan menggunakan kata sistem berkas untuk cabang-cabang yang dilekatkan pada pohon tersebut.

Pada kebanyakan situasi, sistem berkas dilekatkan pada pohon berkas dengan perintah mount. Perintah mount memetakan direktori di dalam pohon berkas yang ada, yang disebut titik mount, ke root dari sistem berkas yang baru. linux mempunyai opsi lazy unmount (umount -l) yang menghapus filesystem dari penamaan hierarki tetapi tidak benar-benar melepaskannya sampai tidak lagi digunakan. umount -f adalah unmount paksa, yang berguna ketika filesystem sibuk. Daripada langsung menggunakan umount -f, kita dapat menggunakan lsof atau fuser untuk mencari tahu proses yang menggunakan filesystem dan menghentikan proses-proses tersebut.



**Organization of the file tree**

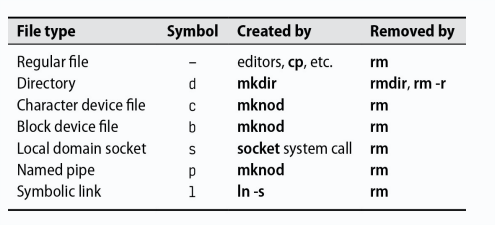
Sistem UNIX tidak pernah terorganisir dengan baik! Berbagai konvensi penamaan yang tidak kompatibel digunakan secara bersamaan, dan berbagai jenis file tersebar secara acak di sekitar ruang nama. Itulah sebabnya mengapa sulit untuk mengupgrade sistem operasi.Sistem berkas root mencakup setidaknya direktori root dan sekumpulan file dan subdirektori minimal. File yang berisi kernel OS biasanya berada di bawah /boot, tetapi nama dan lokasi yang tepat dapat bervariasi. Pada BSD dan beberapa sistem UNIX lainnya, kernel bukanlah sebuah file tunggal melainkan sekumpulan komponen.

/etc berisi file-file sistem dan konfigurasi yang penting. /sbin dan /bin untuk utilitas penting, dan terkadang /tmp untuk berkas-berkas sementara. Secara tradisional /dev merupakan bagian dari sistem berkas root, tetapi sekarang ini /dev merupakan sistem berkas virtual yang di-mount secara terpisah. Beberapa sistem menyimpan berkas-berkas pustaka bersama dan beberapa keanehan lainnya, seperti preprosesor C, di direktori /lib atau /lib64. Sistem lainnya memindahkan berkas-berkas tersebut ke dalam /usr/lib, terkadang meninggalkan /lib sebagai sebuah link simbolik.

/usr dan /var juga sangat penting. /usr adalah tempat penyimpanan program-program standar-tetapi-tidak-kritis-sistem yang paling penting, bersama dengan berbagai berkas lain seperti manual on-line dan sebagian besar pustaka. FreeBSD menyimpan cukup banyak konfigurasi locql di bawah /usr/local. /var menyimpan direktori spool, file log, informasi akuntansi, dan berbagai item lain yang berkembang atau berubah dengan cepat dan bervariasi pada setiap host. Baik /usr dan /var harus tersedia agar sistem dapat masuk ke modus multiuser.



**File types**

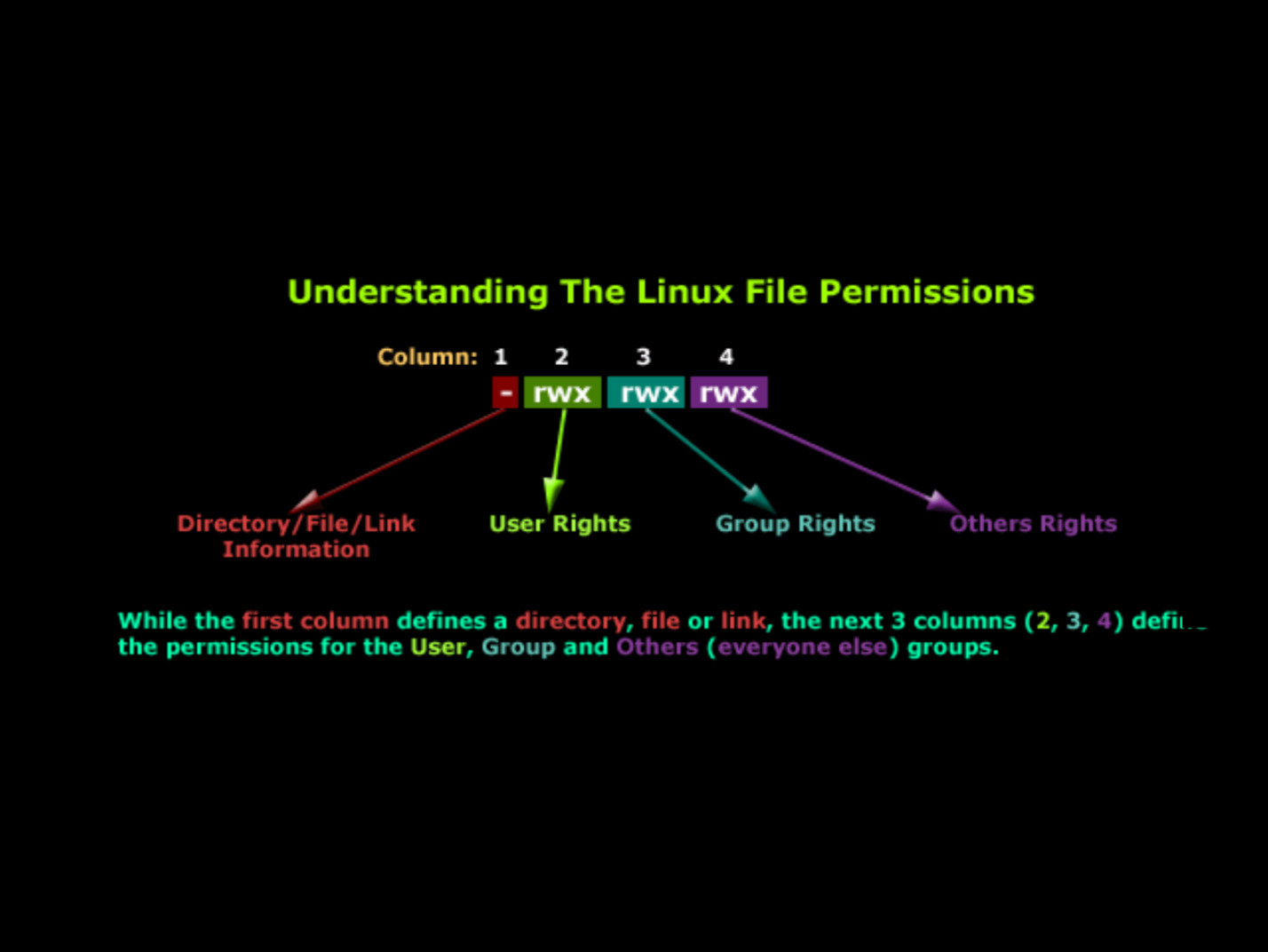


Implementasi filesystem didefinisikan jadi 7 tipe:

* Regular file terdiri dari serangkaian byte; sistem berkas tidak menerapkan struktur pada isinya. File teks, file data, program yang dapat dieksekusi, dan pustaka bersama semuanya disimpan sebagai file biasa.
* Direktori adalah nama referensi ke file lain.
* Character & block device files. Device file memungkinkan program berkomunikasi dengan perangkat keras dan periferal sistem. Kernel menyertakan (atau memuat) perangkat lunak driver untuk setiap perangkat sistem. Perangkat lunak ini menangani detail yang berantakan dalam mengelola setiap perangkat sehingga kernel itu sendiri dapat tetap relatif abstrak dan tidak bergantung pada perangkat keras.Character device memakai aliran karakter untuk komunikasi seperti terminal sedangkan block device menggunakan blok data untuk komunikasi, biasanya seperti hard disk.
* Local domain socket adalah cara bagi proses untuk berkomunikasi satu sama lain. Soket ini mirip dengan soket jaringan, tetapi terbatas pada host lokal. Syslog dan X Window System adalah contoh fasilitas standar yang menggunakan soket domain lokal.
* Named pipes seperti soket domain lokal memungkinkan proses yang sedang berjalan untuk berkomunikasi satu sama lain dalam host yang sama.
* Symbolic links merupakan cara untuk memberikan beberapa nama pada sebuah file, namun lebih fleksibel dibandingkan dengan tautan keras. Tautan ini dapat mengarah ke file pada sistem berkas yang berbeda, dan dapat mengarah ke direktori.

**File attributes**

Pada model sistem berkas Unix dan Linux, setiap berkas memiliki satu set sembilan bit hak akses, yang menentukan siapa yang dapat membaca, menulis, dan mengeksekusi berkas tersebut. Bersama dengan tiga bit lainnya yang terutama mempengaruhi operasi program yang dapat dieksekusi, bit-bit ini merupakan mode file. Kedua belas bit mode ini disimpan bersama dengan empat bit informasi tipe file. Empat bit tipe file ditetapkan ketika file dibuat dan tidak dapat diubah, tetapi pemilik file dan superuser dapat memodifikasi dua belas bit mode dengan perintah chmod.



Bit-bit izin dibagi menjadi tiga kelompok yang masing-masing terdiri dari tiga bit. Kelompok pertama yang terdiri dari tiga bit adalah untuk pemilik berkas, kelompok kedua untuk grup berkas, dan kelompok ketiga untuk semua orang. Anda juga dapat menggunakan notasi oktal (basis 8) karena setiap digit dalam notasi oktal mewakili tiga bit. Tiga bit paling atas (dengan nilai oktal 400, 200, dan 100) mewakili pemilik berkas, tiga bit tengah (dengan nilai oktal 40, 20, dan 10) mewakili grup berkas, dan tiga bit paling bawah (dengan nilai oktal 4, 2, dan 1) mewakili semua orang.

S**etuid (Set User ID)** (s pada permission user)

* Jika diterapkan pada file eksekusi, file akan berjalan dengan hak akses pemilik file, bukan pengguna yang menjalankan.
* Contoh: /usr/bin/passwd menggunakan Setuid agar pengguna biasa bisa mengubah password tanpa akses root.
* Set dengan: chmod u+s file

**Setgid (Set Group ID)** (s pada permission grup)

* Pada file eksekusi: proses berjalan dengan hak grup pemilik file.
* Pada direktori: file baru yang dibuat dalam direktori akan mewarisi grup dari direktori, bukan grup pengguna yang membuat file.
* Set dengan: chmod g+s file/direktori

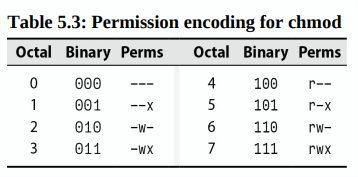
**Sticky Bit** (t pada permission others)

* Digunakan pada direktori publik seperti /tmp, memungkinkan hanya pemilik file atau root yang bisa menghapus/memodifikasi file dalam direktori.
* Set dengan: chmod +t direktori

**Ls: list & inspect files**

Perintah ls mencantumkan daftar file dan direktori. Perintah ini juga dapat digunakan untuk memeriksa atribut file dan direktori. Opsi -l menyebabkan ls menampilkan format panjang, yang mencakup mode file, jumlah tautan keras ke file, pemilik file, grup file, ukuran file dalam byte, waktu modifikasi file, dan nama file.

**Chmod: change permissions**



Perintah chmod mengubah mode dari sebuah file. Anda dapat menggunakan notasi oktal atau notasi simbolik.

Contoh chmod

| specifier | meaning |

|-------|------|-----------|

| u+w | add write permission for the files owner |

| ug=rw,o=r | gives r/w permission to owner & group, and r permission to others |

| a-x | remove execute permission for all users |

|ug=srx,o= | set the setuid, setgid, and sticky bits for owner and group (r/x) |

| g=u | make the groups permission the same as the owners |

Tips: Anda juga dapat menentukan mode yang akan ditetapkan dengan menyalin mode dari file lain dengan opsi --reference. (contoh: chmod --reference = file sumber file target)

**Chown: change ownership**

Perintah chown mengubah pemilik dan grup sebuah berkas. Opsi -R menyebabkan chown mengubah kepemilikan isi berkas secara rekursif.

**Chgrp: change group**

Perintah chgrp mengubah grup sebuah file. Opsi -R menyebabkan chgrp mengubah grup isi file secara rekursif.

**Unmask: set default permissions**

Perintah umask menetapkan izin default untuk file dan direktori baru. Perintah umask adalah sebuah bit mask yang dikurangkan dari izin default untuk menentukan izin yang sebenarnya.

**Access control lists**

Model perizinan Unix tradisional sederhana dan efektif, tetapi memiliki keterbatasan. Sebagai contoh, sulit untuk memberikan sebuah berkas kepada beberapa pemilik, dan sulit untuk memberikan izin yang berbeda kepada sekelompok pengguna pada berkas yang berbeda. Access Control Lists (ACL) adalah cara untuk memperluas model perizinan Unix tradisional. ACL memungkinkan Anda untuk memberikan sebuah file kepada beberapa pemilik dan memberikan izin yang berbeda kepada sekelompok pengguna pada file yang berbeda.

Setiap aturan dalam ACL disebut sebagai entri kontrol akses (ACE). ACE terdiri dari penentu pengguna atau grup, topeng izin, dan tipe. Penentu pengguna atau grup dapat berupa nama pengguna, nama grup, atau kata kunci khusus seperti pemilik atau lainnya. Topeng izin adalah sekumpulan izin, dan jenisnya bisa berupa izinkan atau tolak. Perintah getfacl menampilkan ACL sebuah berkas, dan perintah setfacl menetapkan ACL sebuah berkas.

**Implementasi**

* Posix: jenis acl tradisional yang digunakan pada sistem unix-like seperti linux, FreeBSD dan solaris.

| **Format** | **Example** | **Sets permissions for** |
| --- | --- | --- |
| user::perms | user:rw- | The file's owner |
| user:username:perms | user:abdou:rw- | The user named username |
| group::perms | group:r-x | The file's group |
| group:groupname:perms | group:users:r-x | The group named groupname |
| mask::perms | mask::rwx | The maximum permissions |
| other::perms | other::r-- | Everyone else |

* NFSv4 ACLs: jenis acl yang lebih baru dan canggih, terdapat fitur tambahan seperti acl default yang digunakan untuk menetapkan acl pada file dan direktori baru.