Processer

Introduktion till Linux och små nätverk

Innehåll

1 Vad är en process?

Det är skillnad på ett program och en process. Ett program är ett antal instruktioner som talar om för datorn vad datorn ska göra. En process är utförandet av instruktionerna i programmet under operativsystemets kontroll. Operativsystemets uppgift är att se till att processerna har tillgång till de resurser (ordet resurser används i en bred betydelse och kan vara arbetsminne, lagringsutrymme, optiskt media, skärm m.m.) de behöver, att processerna inte stör varandra och att processer som behöver utbyta information med varandra kan göra det.

Ett exempel: 3 användare är inloggade på en dator. Alla tre har startat webbläsaren. Det innebär att alla tre användarna använder samma webbläsarprogram, men det finns tre olika processer som tar hand om exekveringen av programmet, en process för varje användare.

1.1 Processtabellen

Information om alla processer på ett system finns i *processtabellen*. I informationen ingår vilket konto som startat processen, vilket kommando/program som exekveras, de resurser processorn använder m.m. I processtabellen finns också information om processen är redo (*ready*) eller väntade (*waiting*). En process är redo ifall den är i ett sådant tillstånd att den kan köras direkt när den får tillgång till processorkärnan. En väntade process väntar på någon form av input, t.ex. en tangentbordstryckning eller data från en annan process innan den kan fortsätta exekveringen.

1.2 Processens körning

En CPU, datorns processor, har en eller flera kärnor. Varje processorkärna kan endast exekvera en process i taget. Det gör att så länge en process använder processorkärnan kan ingen annan process använda samma processorkärna. För att inte låsa processorkärnan vid en och samma process under lång tid används *tidsdelning*. Tidsdelning innebär att operativsystemet delar in tiden i delar (*time slices*) vars längd mäts i milli- eller mikrosekunder. Operativsystemet låter sedan varje process använda processorn under en sådan tidsdel. När tidsdelen för en process tagit slut sparas processens tillstånd så att den kan fortsätta där den var nästa gång den får tillgång till processorkärnan. Sedan görs processorkärnan klar för att ta hand om nästa process. Operativsystemet bestämmer hur ofta en process ska få tillgång till processorkärnan med hjälp av prioritering. En högt prioriterad process kommer att få tillgång till kärnan oftare än en lågt prioriterad process.

Att använda tidsdelning ger för användaren ett intryck av att processorn arbetar med flera processer samtidigt, men det är endast en process i taget som finns i processorkärnan. I moderna system har processorn oftast flera kärnor, vilket möjliggör körning av flera processer samtidigt men fortfarande bara en process åt gången per kärna. Processorer med flera kärnor ställer ytterligare krav på operativsystemets hantering av resurser för att undvika störningar och s.k. dödlägen (*deadlocks*). Ett exempel på dödläge är där en process kontrollerar skrivaren och väntar på att få tillgång till DVD-läsaren, medan en annan process kontrollerar DVD-läsaren och väntar på att få tillgång till skrivaren.

Tidsdelning ökar prestandan trots att det krävs en hel del ”administration” för att spara och återskapa processernas tillstånd. I ett system utan tidsdelning skulle processorkärnan vara låst vid en och samma process tills den är klar. Om processen då väntar på någon input kommer hela datorns arbete att stå helt still tills detta input är klart. Med tidsdelning kan processorkärnan arbeta med

annat medan processen väntar på input.

1.3 Processträd

Alla program i en dator exekveras som processer, även det kommandoskal som du använder när du anger textkommandon körs som en process. En körning av ett program från kommandoskalet genererar en ny process, som i sin tur kan skapa ytterligare processer. Varje process (nästan) har alltid en process som kan kallas för processens förälder (*parent*). Barnprocesserna kommer att ärva en del av egenskaperna från föräldraprocessen.

För att visa processernas relationer på ett system kan man använda kommandot `pstree(1)`. Kommandot ger ett träddiagram, ett *processträd*, över processerna på systemet. Vill man se de grenar av processträdet som hör till en viss användare kör man `pstree användarnamn`. Läs manualsidorna eller t.ex. The Linux Information Project [1] för mer information om `pstree` och de växlar som finns tillgängliga.

2 Processhantering

2.1 Kommandot `ps(1)`

Processer kan hanteras på flera sätt. Här kommer vi titta närmare på kommandot `ps(1)`. Körs `ps` i ett terminalfönster visas delar av informationen som finns i processtabellen. Utmatningen kommer att likna tabell 1.

PID	TTY	TIME	CMD
1860	pts/0	00:00:00	bash
1861	pts/0	00:00:00	ps

Tabell 1: Exempel på utmatning från `ps`

Den information som ges är:

`PID`: Processens identitetsnummer. Är unikt för varje process.

`TTY`: Den terminal som processen hör till.

`TIME`: Total CPU-tid för processen.

`CMD`: Kommandot eller programmet som exekveras inom processen. Det är också processens namn. Det kan finnas flera processer med samma namn.

Lägg märke till att `ps` finns med i processtabellen eftersom även körningen av `ps` hanteras som en process.

Processerna som listas är endast de processer som hör till den gren i processträdet som börjar med det kommandoskal som startar processerna.

2.1.1 Några växlar för `ps(1)`

`-f` visar lite mer information om processsen än `ps` utan växlar, bl.a. PPID, *Parent Process ID*, vilket är identifikationsnumret för den process som startade processen med processid `PID`.
`-u` visar mer information om alla processer som startats av ditt användarkonto och som startats via ett kommando i ett terminalfönster.
`-u inloggningsnamn` visar information om alla processer på systemet som startas av `inloggningsnamn`, oavsett om processen startats via ett terminalfönster, via en genväg i det grafiska gränssnittet eller på något annat sätt.
`-fu` visar information om alla processer som startas av ditt användarkonto och som startats via ett kommando i ett terminalfönster och dessutom vilka processer som hör till vilket skal.
`-fu inloggningsnamn` visar utförligare information om alla processer som startats av `inloggningsnamn`, oavsett om processen startats via ett terminalfönster, via en genväg i det grafiska gränssnittet eller på något annat sätt. Här kan du också se den fullständiga sökvägen till kommandot eller programmet som exekveras av processen.
`-ef` visar information om samtliga processer på systemet. Kom ihåg att skicka vidare listan till `more(1)` eller `less(1)` med hjälp av pipe, |, för att lättare kunna läsa listan, t.ex. `ps -ef | less`
`--forest` visar ett processträd över de processer som hör till samma skal som du kör `ps` i.
`-N` negerar valet, d.v.s. döljer de processer som specificerats med växlarna. `-fNU inloggningsnamn` kommer alltså att visa alla processer på systemet som inte startas av `inloggningsnamn`.

Notera att växlarna är känsliga för små och stora bokstäver och i vilken ordning de anges. T.ex. ger `-a` och `-A` olika utdata. Ett annat exempel är `-aU inloggningsnamn` som går bra att använda men `-Ua inloggningsnamn` fungerar inte. Läs mer om grundläggande användning av `ps(1)` i manualsidorna eller t.ex BinaryTides [2]. Kom ihåg att prova själv!

2.2 Processer i förgrunden och i bakgrunden

Se till att alla instanser av webbläsaren Firefox är stängda. Öppna ett terminalfönster och starta Firefox i det terminalfönstret.[1] Webbläsaren startas och visas på skärmen. Titta nu i terminalfönstret.[2] Du kan inte ange några nya kommandon i terminalfönstret förrän webbläsaren, eller snarare processen som hanterar webbläsaren, avslutas. Skälet är att webbläsarprocessen körs i det som kallas förgrunden. Alla processer körs i förgrunden som standard. Om du stänger webbläsarfönstret (och därmed talar om för datorn att den ska avsluta processen) kommer prompten tillbaka i terminalfönstret och du kan ange nya kommandon.

Starta nu webbläsaren med kommandot `firefox &`, d.v.s. lägg till ett mellanslag och &-tecknet[3] innan du trycker `<Enter>`. Webbläsaren startas, precis som tidigare. Byt till terminalfönstret och tryck `<Enter>` så kommer prompten tillbaka. Kör kommandot `ps`. Du bör nu ha (minst) tre processer i listan, `bash`, `firefox` (kan vara något annat ifall du startade en annan webbläsare) samt `ps`. Det du gjorde när du lade till &-tecknet är att du talade om för operativsystemet att köra webbläsarprocessen i bakgrunden direkt för att kunna använda terminalfönstret till annat utan att behöva avsluta webbläsaren. I ett grafiskt gränssnitt är kan användningen av ampersand ses som överflödig. Du kan ju i princip bara starta ett nytt terminalfönster ifall du vill mata in nya kommandon, men anta att du har tillgång till en dator enbart via terminalen (vilket ofta är fallet om det är en server) och vill köra ett stort administrativt jobb eller en omfattande beräkning eller simulering som tar lång tid. Då kan du med hjälp av &-tecknet starta en process som tar lång tid att köra och sedan använda terminalen till annat i väntan på att processen ska bli klar.

För att visa alla pågående jobb används kommandot `jobs`. Läs mer om jobb i kurslitteraturen [3, B.1.4] eller t.ex. Linux Reviews [4].

2.3 Avsluta processer

För att avsluta en process kan man använda kommandot `kill(1)`. `kill(1)` skickar en signal kallad `SIGTERM` som avslutar processen, men låter processen utföra vissa åtgärder, som att lämna tillbaka resurser till systemet, innan den

[1] Du kan prova det här även om du ansluter till din dator via ssh och inte har tillgång till det grafiska användargränssnittet. Om du inte har installerat någon webbläsare kan du starta ett annat program, t.ex. nano.

[2] Ev. dyker några felmeddelanden från GTK eller andra felmeddelanden upp. Det spelar ingen roll för den här övningen.

[3] & kallas ibland ampersand

avslutas. Exakt vad processen gör när den tar emot `SIGTERM` bestäms av hur processen är konfigurerad.

Ett exempel. Starta `nano` i bakgrunden (`nano &`) i ett terminalfönster. Kontrollera att `nano` är igång med `jobs`. Växla till `nano` med kommandot `fg %1`. `fg` flyttar en process till förgrunden. Har du flera jobb igång samtidigt använder du %2, %3, %4 o.s.v. för att välja vilket jobb du vill flytta till förgrunden. Skriv in text i editorn och tryck `Ctrl-z` för att flytta `nano` till bakgrunden igen, utan att spara texten. Ta reda på `PID` för `nano` och avsluta `nano` med kommandot `kill PID` där du byter ut `PID` mot det process-id som `nano` har.

Kör nu `fg %1` för att försöka byta till `nano`. Nu får du ett meddelande om att processen mottog `SIGTERM`, men att den skrivit den osparade text som fanns i programmet till en textfil, `nano.save`. Processen som körde `nano` fick alltså chansen att städa upp efter sig innan den avslutades.

Ibland svarar inte en process på `kill`. Man kan då använda `kill -9 pid` för att skicka signalen `SIGKILL` istället. Processen kommer då att avslutas utan att få chansen att städa upp efter sig. `kill -9` bör endast användas som en sista utväg ifall det inte är möjligt att stänga av processen alls och en omstart av datorn inte är möjlig.

Ett exempel. Starta `nano` i bakgrunden i ett terminalfönster igen. Växla till `nano` (kommandot `fg`), skriv in lite text och flytta `nano` till bakgrunden (`Ctrl-z`). Kör nu kommandot `kill -9 PID` och sedan `fg %1`. Nu får du bara ett meddelande om att processen dödats. Texten som fanns i `nano` har inte sparats undan. Processen fick alltså inte chansen att städa undan efter sig innan den dödades.

Processer kan också avslutas genom att ange processens namn genom kommandot `pkill` med processnamnet som argument, t.ex. `pkill nano`. Detta avslutar alla processer med processnamnet `nano`.

2.4 Andra kommandon och program

Kom ihåg att använda `aptitude(8)` eller `synaptic(8)` för att leta efter och installera andra program för processhantering.

För att visa rader som innehåller ett visst ord kan du skicka listan med alla processer till kommandot `grep(1)`, t.ex. `ps -ef | grep gnome`

`top(1)` visar en automatiskt uppdaterad bild över processerna på systemet, deras minnes- och processoranvändning m.m. ungefär som Aktivitetshanteraren i Windows.

Ett grafiskt användargränssnitt för hantering av processer är `gnome-system-monitor(1)`.

3 Inlämningsuppgift

Under Studieuppgifter finns en uppgift som rör processer och processhantering.

Referenser

[1] The Linux Information Project. How to use the pstree command. [Online]. Available: http://www.linfo.org/pstree.html

[2] BinaryTides. 10 basic examples of Linux ps command. [Online]. Available: https://www.binarytides.com/linux-ps-command/

[3] R. Hertzog and R. Mas, *The Debian Administrator's Handbook, Debian Jessie from Discovery to Mastery*, 1st ed. Lulu. com, 2015.

[4] LinuxReviews. Jobs - the basics of job control. [Online]. Available: http://linuxreviews.org/beginner/jobs/