Processen: oefeningen

Titel	Processen: oefeningen		
Vak	Computer Systems		
OPO Code	MBI07	Versie	2.0

1 Inleiding

Deze oefeningen zijn ook een soort van labo. Voor dit labo is echter niet verplicht te maken en in te dienen. Wat je in dit labo doet is echter wél deel van de leerstof, dus je kan er mogelijk een vraag over verwachten op het examen!

Eén van de belangrijkste taken van een besturingssysteem is het stroomlijnen en inplannen van processen. Een groot deel daarvan doet het besturingssysteem zelf al voor ons, en het zal niet vaak voorvallen dat je veel aan bijvoorbeeld het scheduling algoritme van je OS zal aanpassen. Maar wat is een proces eigenlijk? In dit labo gaan we op verkenning in wat een proces is, welke info we over processen kunnen vinden en hoe we aan de slag kunnen met processen (automatisch) opstarten en beheren.

2 Verkenning met het ps en top commando

2.1 ps

Een zeer veel gebruikt commando in Linux wanneer we over processen bezig zijn is het `ps` commando. In zijn simpelste vorm toont het enkel de processen verbonden aan het huidige terminal-venster. Op dit moment zijn dat er slechts 2: `bash`, het shell-programma dat luistert naar commando's/instructies en `ps`, het commando dat we net hebben uitgevoerd.

```
roel@ubuntu2404-2:~$ ps
    PID TTY          TIME CMD
   4009 pts/2    00:00:00 bash
   4022 pts/2    00:00:00 ps
roel@ubuntu2404-2:~$
```

Om meer info te krijgen kunnen we met opties van dit commando gaan spelen. Een veel gebruikte variatie van dit commando is `ps aux`.

- `a` drukt uit dat de processen van alle gebruikers getoond moeten worden, niet enkel de processen van de huidige gebruiker.
- `u` geeft ons een heleboel meer info per proces, bijvoorbeeld welke gebruiker het gestart heeft, en hoeveel CPU en geheugengebruik het heeft.
- `x` toont ons ook alle processen die niet aan een interactieve terminal verbonden zijn. Je zal merken dat dit er een heleboel zijn, bijvoorbeeld je `NetworkManager`.

```
roel@ubuntu2404-2:~$ ps aux
USER         PID %CPU %MEM    VSZ   RSS TTY      STAT START   TIME COMMAND
root           1  0.1  0.1  23212 14212 ?        Ss   20:59   0:01 /sbin/init splash
root           2  0.0  0.0      0     0 ?        S    20:59   0:00 [kthreadd]
root           3  0.0  0.0      0     0 ?        S    20:59   0:00 [pool_workqueue_release]
root           4  0.0  0.0      0     0 ?        I<   20:59   0:00 [kworker/R-rcu_gp]
root           5  0.0  0.0      0     0 ?        I<   20:59   0:00 [kworker/R-sync_wq]
root           6  0.0  0.0      0     0 ?        I<   20:59   0:00 [kworker/R-kvfree_rcu_reclaim]
root           7  0.0  0.0      0     0 ?        I<   20:59   0:00 [kworker/R-slub_flushwq]
root           8  0.0  0.0      0     0 ?        I<   20:59   0:00 [kworker/R-netns]
root           9  0.0  0.0      0     0 ?        I    20:59   0:00 [kworker/0:0-events]
root          11  0.0  0.0      0     0 ?        I<   20:59   0:00 [kworker/0:0H-events_highpri]
```

Een heleboel van de info die je hier krijgt spreekt voor zich. Per proces vind je één lijn terug met info over dat proces. Je zal merken dat ieder proces een "PID" (Process ID) heeft. Deze PID komt overal terug in het systeem.

Probeer bijvoorbeeld eens:

- “Calculator” te openen
- Met het `ps`-commando de PID van dit nieuwe proces te achterhalen (tip: de executable heet `gnome-calculator`)
- Met het commando `kill` *`PID`* je calculator af te sluiten (vervang *`PID`* hier door de ID die je in de vorige stap hebt achterhaald)

De output van `ps aux` kan nogal overweldigend zijn. Je kan `grep` gebruiken (zie het hoofdstuk over de Linux Command Line) om de output te filteren.

2.1.1 Oefeningen ps

Los de volgende oefeningen op met behulp van het ps-commando en al je andere Linux command line ervaring. De oplossingen staan op de volgende pagina.

Oefening 1: Wat is de PID van je NetworkManager?

Oefening 2: Tel hoeveel processen er actief zijn in de output van `ps aux`. Hiervoor zoek je best eens even op hoe je op Linux het aantal lijnen in je output telt.

Oefening 3: Welk proces gebruikt het meeste CPU%? Tip: je kan met ps --sort sorteren op verschillende kolommen.

Oefening 4: Welk proces gebruikt het meeste MEM%?

2.1.2 Oplossingen ps

Oefening 1:

```
ps aux | grep NetworkManager
```

Oefening 2:

Het volgende commando geeft je een goed idee:

```
ps aux | wc -l
```

Je mag wel niet vergeten dat de header van de tabel dan meegeteld wordt. Je kan dit oplossen met:

```
ps aux --no-header | wc -l
```

Oefening 3:

Je kan de volgende oneliner gebruiken:

```
ps aux --sort pcpu
```

Het laatste proces in deze lijst gebruikt het meest CPU. Je zou met behulp van het commando `tail` enkel de laatste eruit kunnen halen:

```
ps aux --sort pcpu | tail -n 1
```

Oefening 4:

Dit is gelijkaardig aan oefening 3, alleen moeten we hier sorteren op `pmem` in de plaats van `pcpu`:

```
ps aux --sort pmem
```

2.2 top

Naast `ps` bestaat er ook het commando `top`. In grote lijnen doet dit hetzelfde, maar het is dynamischer (wordt automatisch gesorteerd op CPU%, en blijft voortdurend lopen, is geen momentopname).

3 System load

Een ander courant concept over processen en systeemgebruik is de system load van een systeem. Met het commando `uptime` kan je info opvragen over de load van het systeem.

```
tiebevn@compsys:~$ uptime
 13:38:14 up 5 min,  1 user,  load average: 0.62, 0.63, 0.30
tiebevn@compsys:~$
```

Hier krijg je de "load average" over respectievelijk 1, 5 en 15 minuten terug. In andere woorden, in de afgelopen minuut stond het processorgebruik op 0,62 en de afgelopen 5 minuten op 0,63.

Op een single-core systeem zou dit willen zeggen dat er gemiddeld 62% van de tijd een proces actief was op de processor. Echter, de meeste (als niet alle) computers hebben tegenwoordig meerdere cores, en kunnen dus meerdere processen gelijktijdig op deze cores laten lopen.

Op een systeem met 2 cores zou je dus dit percentage moeten delen door 2 (omdat deze load over 2 cores verdeeld is) om het CPU-gebruik te berekenen

Probeer eens een heleboel applicaties (browser vensters, tekstverwerkers, ...) open te zetten en kijk hoe dit je load beïnvloedt.

Meer info over system load vind je hier.

4 Je eigen commando's

In dit deel van het labo gaan we meer high level aan de slag met processen. Wanneer we een commando uitvoeren, gebeurt er op magische wijze van alles. We gaan nu eens proberen onze eigen code als commando's te gebruiken, en eens uitspitten wat er allemaal gebeurt.

In deze GitHub repository vind je een aantal eenvoudige python-scriptjes waarmee je aan de slag kan. Je kan deze op je machine binnenhalen met het volgende commando:

```
git clone https://github.com/UCLL-AO-ComputerSystems/processes-demo-scripts.git
```

Het kan zijn dat het commando `git` ongekend is. Wellicht wil dit zeggen dat je het pakket `git` nog moet installeren.

Wanneer je dit gedaan hebt zal je een nieuwe map vinden met hierin een aantal python-bestanden.

```
tiebevn@compsys:~$ ls
Desktop    Downloads  Pictures  Templates  processes-demo-scripts  student-challenges
Documents  Music      Public    Videos     snap
tiebevn@compsys:~$ cd processes-demo-scripts/
tiebevn@compsys:~/processes-demo-scripts$ ls
cards.py                count-vowels.py  fibonacci.py    random-number.py
celcius-to-farenheit.py  countdown.py     hello-world.py
tiebevn@compsys:~/processes-demo-scripts$
```

Je kan een bestand in principe uitvoeren door er `./` voor te zetten en proberen uit te voeren als een commando, bijvoorbeeld `./random-number.py`. Je zal echter heel snel merken dat dit ons een error geeft. Dit komt omdat we geen rechten hebben om bestanden uit te voeren.

In ons geval zijn de bestanden niet e**x**ecutable. Dat kunnen we afleiden uit de output van `ls -la`. We kunnen dit aanpassen met een `chmod` commando, bijvoorbeeld:

```
chmod a+x random-number.py
```

Dit commando zet de execute bit aan voor alle gebruikers.

Je kan `chmod` ook voor alle bestanden in een map uitvoeren. Hiervoor gebruik je de optie `-R` (recursive) en verwijs je naar de huidige map:

```
chmod a+x -R .
```

```
tiebevn@compsys:~/processes-demo-scripts$ ls -la
total 40
drwxr-xr-x  3 tiebevn tiebevn 4096 Oct 23 14:27 .
drwxr-x--- 19 tiebevn tiebevn 4096 Oct 23 14:24 ..
drw-r--r--  8 tiebevn tiebevn 4096 Oct 23 14:24 .git
-rw-r--r--  1 tiebevn tiebevn  353 Oct 23 14:24 cards.py
-rw-r--r--  1 tiebevn tiebevn  294 Oct 23 14:24 celcius-to-farenheit.py
-rw-r--r--  1 tiebevn tiebevn  423 Oct 23 14:24 count-vowels.py
-rw-r--r--  1 tiebevn tiebevn  292 Oct 23 14:24 countdown.py
-rw-r--r--  1 tiebevn tiebevn  588 Oct 22 17:16 fibonacci.py
-rw-r--r--  1 tiebevn tiebevn   83 Oct 23 14:24 hello-world.py
-rw-r--r--  1 tiebevn tiebevn  151 Oct 23 14:24 random-number.py
tiebevn@compsys:~/processes-demo-scripts$ chmod a+x -R .
tiebevn@compsys:~/processes-demo-scripts$ ls -la
total 40
drwxr-xr-x  3 tiebevn tiebevn 4096 Oct 23 14:27 .
drwxr-x--- 19 tiebevn tiebevn 4096 Oct 23 14:24 ..
drwxr-xr-x  8 tiebevn tiebevn 4096 Oct 23 14:24 .git
-rwxr-xr-x  1 tiebevn tiebevn  353 Oct 23 14:24 cards.py
-rwxr-xr-x  1 tiebevn tiebevn  294 Oct 23 14:24 celcius-to-farenheit.py
-rwxr-xr-x  1 tiebevn tiebevn  423 Oct 23 14:24 count-vowels.py
-rwxr-xr-x  1 tiebevn tiebevn  292 Oct 23 14:24 countdown.py
-rwxr-xr-x  1 tiebevn tiebevn  588 Oct 22 17:16 fibonacci.py
-rwxr-xr-x  1 tiebevn tiebevn   83 Oct 23 14:24 hello-world.py
-rwxr-xr-x  1 tiebevn tiebevn  151 Oct 23 14:24 random-number.py
```

Wanneer we de x-permission op true hebben gezet, zou je elk bestand hier moeten kunnen uitvoeren door *`./bestandnaam.py`* uit te voeren.

4.1 Path

Natuurlijk werkt dit momenteel enkel als we in de map van dit bestand zitten. Moesten we een map hoger zitten, zou het nog wel werken om `./processes-demo-scripts/count-vowels.py` nog wel werken, maar het wordt snel heel omslachtig om telkens te onthouden welk bestand waar staat.

Hiervoor kunnen we gebruik maken van ons path. Het path, soms ook wel eens het search path of system path genoemd, is een serie van mappen waarin de shell automatisch naar bestanden gaat zoeken. Dit path is automatisch voorgeprogrammeerd, en kan je opvragen door het commando `echo $PATH`.

```
tiebevn@compsys:~/processes-demo-scripts$ echo $PATH
/usr/local/sbin:/usr/local/bin:/usr/sbin:/usr/bin:/sbin:/bin:/usr/games:/usr/local/games:/snap/bin:/snap/bin
tiebevn@compsys:~/processes-demo-scripts$
```

Je krijgt hier een opsomming van een heleboel mappen. Laten we eens 1 van deze mappen gaan bekijken. Bekijk eens welke bestanden er allemaal in de map `/usr/bin` zitten. Je zal hier een ruwe 1000-2000 bestanden tegenkomen die je op het eerste gezicht misschien niet veel zeggen, maar als je wat rond scrolt kom je bestanden tegen als `cd`, `ls`, `pwd`, `echo`, … Allemaal commando's die je al kent. Dit zijn namelijk de bestanden met de code die nodig is om bepaalde commando's uit te voeren, en wanneer je bijvoorbeeld het commando `ls` uitvoert, wordt er teruggegrepen naar het bestand dat hier staat.

De inhoud van `/usr/bin` wordt beheerd door de package manager (`apt`), dus daar kunnen we maar beter afblijven. Een andere directory op ons zoekpad is echter `/usr/local/bin`. Hier kunnen we onze eigen programma's in zetten. Laten we eens proberen om een van onze python-scripts in deze map te zetten.

```
sudo cp cards.py /usr/local/bin/cards
```

Probeer hierna eens het commando `cards` uit te voeren. In principe zou je nu een nieuw commando moeten hebben gemaakt! Ook vanuit andere mappen en users kan je hier in principe aan. Probeer dit nu ook eens voor de andere scriptjes.

Het omgekeerde is natuurlijk ook waar. Als je bijvoorbeeld het bestand `/bin/zip` een andere naam geeft zal het commando `zip` niet meer bestaan.

5 Services/SystemD

Niet alle processen die op je besturingssysteem draaien, hebben we zelf opgestart, en sommige processen willen we graag automatisch laten opstarten. Hiervoor kunnen we gebruik maken van services.

5.1 Op verkenning in services

Een goed voorbeeld is een webserver als nginx. Dit proces draait in de achtergrond, en start ideaal gezien vanzelf op wanneer de server opstart. Daarom wordt er voor nginx bij de installatie een service aangemaakt, die er voor zorgt dat nginx bij het opstarten van het systeem mee opgestart wordt.

Als je `nginx` nog niet hebt geïnstalleerd kan je dat met `apt` doen:

```
sudo apt install nginx
```

Je kan de info over deze service terugvinden met het commando `systemctl status nginx`. Hier krijg je info over het proces, wat er eventueel gelogd wordt, ... Je kan services zelf starten en stoppen met bijvoorbeeld `systemctl stop nginx` en `systemctl start nginx`.

5.2 Je eigen service

Laten we zelf zo eens een service aanmaken. Een service dient altijd in een bestand beschreven te worden in een `.service`-bestand. Dit bestand beschrijft de service, onder welke user het opgestart wordt, welk commando er in de achtergrond uitgevoerd moet worden, welke services op voorhand al in orde moeten zijn ... (het heeft weinig zin om een netwerkservice te maken voor de netwerkmanager werkt bijvoorbeeld.)

5.2.1 Voorbereiding

Eerst en vooral zal je de code moeten binnenhalen en een plaats geven in je systeem.

```
git clone https://github.com/UCLL-AO-ComputerSystems/simple-python-webservice.git
```

Bovenstaand commando zal de repository binnenhalen met alle nodige code. Hierna verplaatsen we de map die we net hebben binnengehaald naar `/opt/simple-service`. Je zou vanuit het hoofdstuk over de Linux command line nog moeten weten hoe dat moet.

Wanneer we dit goed doen zouden we de server manueel moeten kunnen opstarten. Wanneer dit lukt (zie output hieronder) kan je in je browser (van je VM) naar `http://localhost:8000` surfen en zie je de website.

Wanneer dit gelukt is gaan we dit proces proberen te automatiseren. Je kan het actieve commando afsluiten met CTRL+C. De volgende stap is een service te maken die dit voor ons gaat doen.

5.2.2 De service aanmaken

Iedere service wordt beschreven in een service file. Je vindt er een heleboel terug in de map `/etc/systemd/system/`. Wij gaan hier onze eigen file bijvoegen:

```
[Unit]
Description=Simple Python Service
After=network.target
StartLimitIntervalSec=0

[Service]
Type=simple
Restart=always
RestartSec=1
User=tiebevn
ExecStart=/usr/bin/python3 /opt/simple-service/main.py

[Install]
WantedBy=multi-user.target
```

Sla dit bestand op in `/etc/systemd/system/simple-service.service`.

In dit bestand staat een heleboel, enkele van de belangrijkste dingen zijn:

- **Description:** een kleine beschrijving van wat de service doet
- **After:** één of meerdere services die opgestart moeten zijn voordat deze service opgestart wordt.
    - Dit is belangrijk als je met dependencies zit. In dit geval starten we een kleine webserver op die op het lokale netwerk beschikbaar is. Als eerst de netwerkadapter van de machine niet opgestart is kunnen we niet veel doen, vandaar dat we wachten op `network.target`.
- **User:** alles wordt altijd als een bepaalde gebruiker uitgevoerd, die wordt hier meegegeven.
    - Je zal in dit bestand je **eigen username** moeten instellen.

- **ExecStart:** het commando dat bij het opstarten van de service uitgevoerd wordt.

Herlaad nu `systemd` met het volgende commando:

```
sudo systemctl daemon-reload
```

Zo doe je `systemd` veranderingen in `.service` files herladen.

Als je dit goed hebt gedaan kan je nu een nieuwe service met de naam `simple-service` opstarten (zie de commando's een beetje hierboven over `nginx`) en kan je in je browser terug naar de website.

Om de service aan te maken zal je waarschijnlijk een command line tekstverwerker zoals `nano` moeten gebruiken.

Niet eender welke user kan in deze map iets aanpassen. Je zal wellicht ook administrator moeten zijn, of gebruik maken van het `sudo`-commando om dit goed te kunnen doen.

Je gaat een file als deze niet vanbuiten moeten kunnen uitschrijven, ook de lectoren (althans de lector die dit schrijft) kennen deze syntax ook niet vanbuiten. Het kan wel zijn dat we je een gelijkaardig bestand geven en vragen om iets aan te passen of een foutje eruit te halen. Maak zeker ook gebruik van `man systemd.service`!

5.3 Oefeningen

Probeer de volgende oefeningen te maken. De oplossingen vind je op de volgende pagina.

5.3.1 Opgaves

Oefening 1: services automatiseren

Probeer je VM eens opnieuw op te starten. Werkt je webserver nog wanneer de VM terug is opgestart? Wellicht niet, omdat we de service nog niet automatisch laten opstarten. Er is een optie die je kan meegeven met het `systemctl`-commando waarmee je dit heel eenvoudig kan doen, zoek deze op en zorg dat je service bij het herstarten van je VM automatisch opstart.

Oefening 2: service user

Momenteel voeren we dit proces uit als onze eigen user. Dit is niet ideaal, en in de meeste gevallen wil je voor cases zoals dit een service user aanmaken. Probeer een nieuwe user aan te maken, en pas vervolgens de service aan zodat de service als deze user opgestart wordt.

5.3.2 Oplossingen

Oefening 1:

Dit kan met de `enable`-optie van `systemctl`:

```
systemctl enable simple-service
```

Oefening 2:

Je kan een nieuwe service user aanmaken (bijvoorbeeld `simple-service-user`) met het volgende commando:

```
sudo useradd -r -s /bin/nologin simple-service-user
```

`-s /bin/nologin` zorgt ervoor dat er voor deze user geen login shell aangemaakt wordt. Met andere woorden: niemand kan als deze gebruiker aanmelden. Deze gebruiker dient immers enkel om de service te runnen.

Vervolgens kan je in het `.service`-bestand de `User=`-lijn aanpassen om een andere user te gebruiken.