FACULTEIT INGENIEURSWETENSCHAPPEN
EN ARCHITECTUUR

Besturingssystemen II: labo

W. Van den Breen

Bachelor in de industriële wetenschappen: informatica
Academiejaar 2019–2020

UNIVERSITEIT
GENT

Deel I: Inleiding

Lokaal inloggen op een Linux toestel kan op twee manieren gebeuren. Je kan inloggen op de grafische console of je kan inloggen op een van de vijf a zes tekstuele consoles. Veranderen van console kan via de toetsencombinaties CTRL-ALT-F1, CTRL-ALT-F2, … , CTRL-ALT-F7. Wanneer je Linux draait m.b.v. VirtualBox, dan moet je mogelijks gebruikmaken van de toetsencombinaties HOSTKEY-F1, HOSTKEY-F2, …, HOSTKEY-F7 waarbij de HOSTKEY standaard is ingesteld op de rechter CTRL-toets. Na het opstarten van Linux wordt aan iedere console een tty (terminal type) gekoppeld of in het geval van een grafische console een login manager zoals bv. gdm (GNOME Display Manager).

Bij Fedora Linux en aanverwante distributies zal de eerste console ingenomen worden door de grafische console. De overige tekstuele consoles vind je door te switchen met CTRL-ALT-Fx. Voor Debian, Ubuntu, … zal je merken dat de grafische console terug te vinden is op de zevende terminal.

Inloggen op meerdere terminals is mogelijk. Wanneer je correct inlogt op een tekstuele console kom je terecht in een BASH-shell. Bij het inloggen op de grafische console wordt naar alle waarschijnlijkheid een desktopomgeving zoals bv. GNOME, KDE, XFCE, … gestart. In die desktopomgeving kan je dan allerhande GUI-gebaseerde programma's starten naar analogie met MS Windows.

Voor deze labo's is het gebruik van een BASH-shell noodzakelijk. Op een tekstuele console beschik je onmiddellijk na het inloggen over een BASH-shell en op een grafische console zal je een terminal-emulator moeten starten.

1. De terminal

Log in als gebruiker *root* met als wachtwoord *e=mc**2* en start wanneer van toepassing in GNOME de terminal-emulator. Tik in het venster de opdracht "**stty -a"** in.

1. Welke toetsencombinatie heb je nodig om een runnend programma te onderbreken?
2. Hoe kan je een runnend programma pauzeren? (het hervatten van een programma zullen we later zien)
3. Wanneer een programma vraagt om gegevens in te typen, met welke toetsencombinatie kan je dan aangeven dat de invoer stopt?

Je kan vraag 1 en vraag 3 het best uitproberen m.b.v. het commando "**cat**". Deze opdracht kopieert alle invoer van het toetsenbord (standaard invoer) naar het scherm (standaard uitvoer).

Probeer dit nu ook uit met de opdracht "**mail root@localhost**". Hier zal eerst gevraagd worden om het onderwerp van het bericht in te geven, gevolgd door het bericht zelf. Wanneer je het einde van de invoer aangeeft, zal een mail verstuurd worden naar de gebruiker root op het lokale toestel. De gebruiker root kan zijn mail lezen door de opdracht **mail** uit te voeren zonder parameters. Hierdoor kom je terecht in een interactieve omgeving waar je een overzicht krijgt van alle berichten. Bij het intikken van een nummer dat naast een bericht staat, kan je de mail lezen. Het bericht verwijderen kan door het ingeven van de letter d gevolgd door het berichtnummer. De interactieve omgeving verlaten kan door het ingeven van de letter **q**.

4. Wanneer je op de commandolijn een aantal woorden intikt, hoe kan je dan het laatste woord verwijderen?

5. In een shell kan je aan "auto completion" doen door gebruik te maken van de tab-toets. Tik de letters "les" in, en tik vervolgens op de tab-toets. Je merkt dat er automatisch less verschijnt. Nu kan het zijn dat er nog opdrachten zijn die met less beginnen. Tik nogmaals op de tab-toets om te kijken welke opdrachten er met less beginnen. Een ander handigheidje is "reverse search" om eerder ingetypte commando's te suggereren tijdens het typen. Gebruik hiervoor "CTRL+r".

6. Linux telt enkele duizenden opdrachten met elk nog een variabel aantal opties. De opdracht zelf wordt doorgaans beperkt tot enkele letters die de afkorting vormen van een woord dat de functie van de opdracht deels omschrijft. Zo staat het commando "**ls"** voor "list**"**, "**mkdir**" voor "make directory", het commando "**wc"** voor "word count**"**, het commando "**cc"** voor "C-compiler**"**, het commando "**dd**" voor "convert and copy**"**. Bij het laatste voorbeeld is er een conflict omdat de afkorting cc reeds gebruikt wordt voor de C-compiler. Als oplossing heeft men de volgende letter van het alfabet gebruikt.

   "Auto completion" voor commando's werd reeds in het vorige punt naar voor gebracht maar geldt ook voor de bijhorende opties. Geef op de commandolijn het commando "**ls -**" in en gebruik vervolgens de tab-toets. Je merkt dat er een pak opties verschijnen die voorafgegaan worden door twee mintekens. Onder Linux kan je veelal opties op de commandolijn meegeven in twee formaten. Het korte formaat bestaat uit één letter en wordt voorafgegaan door één minteken. Dezelfde optie kan je ook meegeven d.m.v. het lange formaat. In dat geval zal de

optie voorafgegaan worden door twee opeenvolgende mintekens. Wanneer je de keuze hebt tussen de twee formaten zal je merken dat de korte variant de voorkeur geniet.

Zo kan het commando "**wc --words --lines /etc/passwd**" ook geschreven worden als "**wc -w -l /etc/passwd**" en zelfs als "**wc -wl /etc/passwd**". De opdracht toont het aantal lijnen en het aantal woorden uit het bestand /etc/passwd.

2. Gebruik van enkele eenvoudige opdrachten

    1. Met de opdracht man kan je alle info over een welbepaalde opdracht achterhalen. Zo zal de opdracht "**man ls**" de info tonen van de ls-opdracht.

       Man toont de informatie a.d.h.v. een pager, een programma dat de informatie niet alleen op het scherm toont maar dat ook gebruikersinteractie toelaat. Zo kan je scrollen, zoeken naar bepaalde woorden, etc. Standaard is de pager ingesteld op het commando **less**. Dit betekent dat wanneer je wil weten hoe je voorwaarts moet zoeken in een manpagina, je dit moet gaan zoeken bij het commando less en niet bij het commando man. Vraag de manpagina op van het commando less en ga na hoe je een tekst die met less wordt getoond kan afsluiten. Hoe kan je voorwaarts zoeken in een manpagina?
    2. Hoe kan je zorgen dat er bij het zoeken in een manpagina geen rekening wordt gehouden met het verschil tussen hoofd- en kleine letters?
    3. Bekijk de manpagina van het commando man en ga na hoeveel secties er gekend zijn.
    4. Wanneer je de opdracht "**man read**" intikt krijg je de info te zien van het commando read. Er is echter ook een systeemaanroep read aanwezig. Hoe kan je aan man meegeven dat je niet de info wenst te zien van het commando read maar wel van de systeemaanroep read?
    5. Met de opdracht "**ls"** krijg je van een directory een overzicht van alle bestanden en subdirectories te zien. Wanneer je geen directory opgeeft, wordt de huidige werkdirectory genomen. Wat doen de opties **-l** en **-h**?
    6. Bekijk met "**ls /"** de inhoud van de hoofddirectory.
    7. In punt 5 en 6 heb je gemerkt dat de uitvoer voorzien wordt van kleuren. Zo worden bv. directories in het donkerblauw gekleurd en symbolische links in het lichtblauw. Dit gedrag is te wijten aan het feit dat er voor ls een alias gedefinieerd is die telkens ls vervangt door "**ls --color=auto**". Ga met de opdracht "**alias**" na welke andere aliassen er bestaan.
    8. Met de opdracht "**cd**" kan je van werkdirectory veranderen. Wanneer je geen directorynaam opgeeft wordt je home-directory de nieuwe werkdirectory. Voer "**cd /tmp**" uit en keer daarna terug naar je home-directory.
    9. Een directory aanmaken kan via de opdracht "**mkdir**". Maak in je home-directory een directory aan met als naam 'c' waar je toekomstige C-programma's naartoe kan kopiëren.

10. Een bestand kopiëren gebeurt via de opdracht “**cp**”. De eerste parameter is de bron, de tweede de bestemming. Een bestand verplaatsen doe je via de opdracht “**mv**”.

Deel II: Compileren in de Shell

In Linux is het vrij eenvoudig om een C of C++ programma te compileren zonder gebruik te maken van een IDE. Het enige wat je nodig hebt is een eenvoudige tekstverwerker. In een grafische omgeving kun je bvb. **gedit** gebruiken. In tekstuele consoles is **nano** eenvoudig te gebruiken, terwijl **vi**, **vim**, **emacs**, … al wat geavanceerder zijn.

Open in BASH een tekstverwerker naar keuze en tik daarin onderstaand codefragment:

```
#include <stdio.h>

int
main(){
        printf("Hello world!");
        return 0;
}
```

Sla dit bestand vervolgens op als hello.c (let op de extensie). Om nu dit bestand te compileren voer je op de commandolijn "**cc hello.c -o hello**" of "**make hello**" uit. Na het succesvol compileren kan je bestand uitvoeren door op de commandolijn "**./hello**" in te tikken. Bemerk dat je voor het uitvoerbaar bestand de "./" niet mag vergeten!

Wanneer je met de opdracht "**file**" informatie over het uitvoerbaar bestand opvraagt, zal je merken dat er gebruikgemaakt wordt van **shared libraries** en dat het uitvoerbaar bestand dus niet alle code bevat om de string "Hello world!" naar het scherm te schrijven. Is dit niet de bedoeling, dan kan je bij het compileren de compileroptie "**-static**" opgeven. Hierdoor wordt alles statisch gelinkt en krijg je dus een uitvoerbaar bestand dat alle code bevat om de gegeven tekst naar het scherm te schrijven.

Opdracht

Compileer bovenstaand bestand dynamisch en statisch en bekijk het verschil in grootte. De grootte van een bestand kan je het gemakkelijkst achterhalen met de opdracht "**du**" of met de opdracht "**ls**". Maak in beide gevallen handig gebruik van de optie "-h" om de bestandsgrootte in bytes, KB, MB, … te krijgen.

Deel III: Profiler programma's

Bij het schrijven van programma's zijn er een aantal zaken waar je toch moet op letten. Zo mag bv. de geschreven code geen memory leakage vertonen, mag de uitvoeringstijd niet te groot zijn en moet het aantal systeemaanroepen tot een minimum worden beperkt. Onder Linux zijn er een aantal profiler programma's waarmee je uitvoerbare bestanden kan onderzoeken.

Valgrind

Valgrind is een tool die in zijn meest eenvoudige vorm onderzoekt of de geschreven software memory leakage vertoond.

Opdracht

Schrijf een C-programma dat via malloc een tabel van 2000 gehele getallen aanmaakt en deze getallen daarna gewoon uitschrijft. Je hoeft bewust het in beslag genomen geheugen niet vrij te geven. Compileer het programma en voer het daarna uit d.m.v. "**valgrind ./prog**". Herschrijf nu het programma waar je het gealloceerde geheugen vrijgeeft en controleer opnieuw met valgrind of het programma nog geheugenlekken vertoond.

Time

Om na te gaan hoeveel CPU-tijd een bepaald programma in beslag heeft genomen kan je de opdracht "**time**" gebruiken. Het commando time geeft standaard drie timestamps terug. Het betreft, de totale tijd tussen het starten en het stoppen van het proces, de CPU-tijd voor het uitvoeren van code in USER-mode en tot slot de CPU-tijd voor het uitvoeren van code in de KERNEL-mode. De allereerste tijdsaanduiding is eigenlijk de minst interessante omdat een proces doorgaans niet in één keer wordt uitgevoerd maar door het besturingssysteem verschillende keren wordt onderbroken.

Opdracht

Herneem het programma dat je daarnet hebt geschreven en voer het nu uit d.m.v. "**time ./prog**".

Strace

Soms kan het interessant zijn om te weten welke systeemaanroepen een programma gebruikt. Bekijk de uitvoer van "**strace cat /etc/passwd**". Dit geeft een sequentieel overzicht van alles wat bij het uitvoeren van "cat /etc/passwd" gebeurt. Wanneer je een overzicht wil krijgen van welke systeemaanroepen er gebruikt worden en hoeveel keer ze werden opgeroepen, dan kan je aan strace de optie "**-c**" meegeven.

Deel IV: Een eerste programmeeropdracht

Er zijn een aantal commando's die "ls" als prefix hebben en die allemaal wel de inhoud van "iets" naar het scherm schrijven. Net zoals **ls** de inhoud van een directory toont, toont **lspci** de aanwezige PCI(e)-apparaten, toont **lsusb** de aanwezige USB-apparaten en toont **lsmod** de modules, of drivers, die momenteel door het systeem gebruikt worden.

Opdracht

De bedoeling is een eigen versie van het commando "**lspci -n**" te programmeren in de programmeertaal C. Dit commando overloopt alle mogelijke PCI-adressen en gaat na of er zich een apparaat bevindt. Bevindt er zich een apparaat, dan wordt het adres (busnummer, devicenummer en functienummer) naar het scherm geschreven, samen met het vendorID en het deviceID.

Info en werkwijze

Om een register van een bepaald apparaat, aangesloten op de PCI-bus, aan te spreken moet je te beschikken over het busnummer, devicenummer, functienummer en tot slot het registernummer.

Het aantal bussen binnen het PCI-systeem kan oplopen tot 256. Om te zoeken naar hardware is het dus aangewezen om alle mogelijke bussen af te lopen.

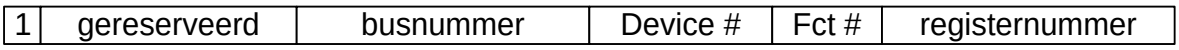
Een device, of apparaat is een fysiek aangesloten “ding” op het PCI-systeem. Voorbeelden hiervan zijn videokaarten, geluidskaarten, onderdelen van de chipset, netwerkadapters, etc. Iedere bus kan theoretisch 32 apparaten bevatten.

Ieder apparaat heeft ten minste één functie en ieder aangesloten apparaat op de PCI-bus kan maximum 8 functies hebben, genummerd van 0 tot 7.

Iedere functie van een apparaat heeft 256 registers. Registers 0 tot 3F bevatten een zee aan informatie over een bepaalde functie van een bepaald apparaat. Registers 40 tot FF bevatten dan weer merkspecifieke informatie.

Registers 0 en 1 bevatten het vendorID en registers 2 en 3 bevatten het deviceID. Wanneer bijvoorbeeld het vendorID 8086 is en het deviceID 2829 kan je vrij snel achterhalen dat het om een SATA AHCI controller gaat van Intel. Deze informatie kan je terugvinden in de PCI vendor database (http://www.pcidatabase.com).

Om de PCI-bus af te scannen moet je telkens een 32-bit getal naar het indexregister (adres = 0xcf8) van het PCI-systeem sturen. Dit 32-bit getal heeft steeds de volgende structuur:



31 0

Je laat het busnummer variëren van 0 tot 255 (8 bit), het devicenummer telkens van 0 tot 32 (5 bit) en ten slotte het functienummer van 0 tot 7 (3 bit). Het registernummer (8 bit) stel je in op 0.

Het antwoord krijg je door vervolgens een 32-bit getal te lezen van het dataregister (adres = 0xcfc). Indien het antwoord uitsluitend 1-bits telt (0xffffffff), is er op die plaats geen apparaat aanwezig. In het andere geval ontvang je de inhoud van de eerste vier registers (registers 0 tot 3) van de geadresseerde functie. De meest significante 16-bits bevatten het deviceID en de minst significante 16-bits het vendorID.

Opmerkingen:

- Om rechtstreeks de hardware te kunnen benaderen, moet je in je code daar expliciet toestemming voor vragen en ook voor krijgen. Bekijk de man-pagina's van de systeemaanroepen **iopl** en **ioperm**.
- Net zoals de meeste systeemaanroepen geven deze functies -1 terug wanneer er iets foutloopt en wordt de gehele variabele **errno** ingesteld. Om de foutboodschap die bij **errno** hoort naar het scherm te schrijven gebruik je de functie **perror(…)** die naast de string die als parameter wordt meegegeven ook de stringversie van errno op het scherm toont. Gebruik dus voor het tonen van foutboodschappen van systeemaanroepen steeds de perror-functie en maak dus geen gebruik van printf.
- Om een 32-bit getal naar een adres te schrijven kan je gebruikmaken van de systeemaanroep "**outl**". Een 32-bit getal van een adres lezen kan via "**inl**". Bekijk de manpagina van outl/inl voor de juiste syntax. Opgelet, inl en outl zijn ook gewone Linux-commando's. Zorg er dus voor dat je de juiste man-pagina oproept.