IT-Universitetet i København 21. august 2014

Operativsystemer og C

Ugeseddel 1 - (uge 35) - 26/8-1/9

Forelæsning: Fredag d. 29. august kl. 10.00 - 11.50 i Aud 3. (2A56)

Emner

Introduktion til kurset. Hvad er et operativsystem? Et moderne computersystem. Bootstrap. Computersystem arkitektur. Samspil mellem operativsystem og hardware. User mode og kernel mode, interrupts og traps. Programmering i C under Linux.

Litteratur og andet kursusmateriale

Silbersschatz, kapitel 1, afsnit 1.1-1.5, 1.13.

(Kernighan&Ritchie, afsnit 1.1-1.10, 5.10, 7.2-7.4).

Online (gratis) materiale om C programmering, som dækker de gennemgåede emner.

- Brian W. Kernighan, Dennis M. Ritchie: The C Programming Language, th Edition Prentice Hall. 1988 http://freecomputerbooks.com/The-C-Programming-Language.html
- The C Book, 2nd edition by Mike Banahan, Declan Brady and Mark Doran, originally published by Addison Wesley in 1991. http://publications.gbdirect.co.uk/c_book/.
- C for Java Programmers, Jason Maassen, VU University, Amsterdam. http://faculty.ksu.edu.sa/jebari_chaker/papers/C_for_Java_Programmers.pdf
- Programmering i C, Kurt Nørmark, Institut for Datalogi, Aalborg Universitet. http://people.cs.aau.dk/ normark/c-prog-06/html/notes/theme-index.html

Mange andre tutorials og kildekodeeksempler kan findes på www.-siderne, f.eks;

- The GNU C Programming Tutorial. http://crasseux.com/books/ctutorial/

Øvelser: Fredag d. 29. august kl. 12.00 - 13.50 i 4A16.

Forberedelser (lav dette før øvelserne går i gang)

*** Til de allerførste øvelser kan I starte med at lave forberedelserne ***

Til øvelserne i denne uge skal du bruge

- Adgang til Linux system.
- C compiler `gcc` og en tekst editor.

Hvis du anvender Windows på din bærbar, har du mindst to (gratis) muligheder for at få adgang til et Linux system, hvor den ene er:

1. Du kan logge på `ssh.itu.dk`, der kører Linux og anvende `gcc`. Installer f.eks. SSHSecureShellClient - se vejledning på intranet https://intranet.itu.dk/en/Intranet-hjem/Organisation/Afdelinger/It-afdelingen/IT-Afdelingens-ABC/SSH.

Hvis du anvender Linux på din bærbar og ikke har installeret `gcc` på dit system, så er det på høje tid. Hvis du anvender Mac OS kan du logge på `ssh.itu.dk` ved at åbne en terminal og bruge `ssh`. Det er ikke nok at installere `Xcode` (som indeholder `gcc`), da vi skal programmere op mod Linux operativsystemets grænseflade.

Oversættelse af C programmer.

Følgende viser meget kort hvordan du kan oversætte C-programmer med `gcc`, som er GNU's C-compiler, ved brug af en Linux shell.

For at ovesætte C kildekode (en `.c`-fil) til en objektfil (en `.o`-fil) skriver du:

```
gcc -c filename.c
```

For at linke flere objektfiler sammen til en eksekverbar fil skriver du:

```
gcc -o programnavn filename1.o filename2.o ...
```

For at køre programmet skriver du derefter:

```
./programnavn
```

Oversættelse af kildekode og linkning af objektfiler kan også gøres på én linje:

```
gcc -o programnavn filename1.c filename2.c filename3.o ...
```

Opgave 1. Mine første C programmer ($\frac{1}{2}$ time)

Dette er begynderopgaver i C programmering (hvis du allerede har erfaring med C programmering, samt ved hvordan du kan oversætte og køre et C program på din maskine, kan du springe denne opgave over og fortsætte nedefor).

Målet med denne opgave er, at du kan

- Skrive, oversætte og køre C programmer.
- Indlæse argumenter fra kommandolinjen, skrive til stdout.

1.1) Skriv et `HelloWorld` program i C. Oversæt programmet og kør det.

1.2) Skriv et program i C, der tager en tekststreng som input argument fra kommandolinjen og udskriver denne på skærmen. Oversæt programmet og kør det.

Opgave 2. Match en tekststreng (min egen grep) (1 time)

Målet med denne opgave er, at du kan

- Indlæse argumenter fra kommandolinjen og skrive til stdout.
- Få øvelse i anvendelse af standard biblioteker, f.eks. `<stdio.h>`, herunder indlæse fra en fil og sammenligne tekststrenge.

2.1) Skriv et program i C, der tager en tekststreng og et filnavn som input argumenter fra kommandolinjen, indlæser filen og finder alle de linjer der indeholder tekststrengen. Alle de fundne linjer skal udskrives på skærmen.

Hint: I filen `observer.c` (se næste opgave) kan du få inspiration til hvordan man kan åbne og hente linjer i en fil. Anvend desuden `<string.h>` (du kan få information om dette bibliotek ved `man string`).

Opgave 3. `/proc/cpuinfo` og observer.c (1 time)

Baseret på en opgave af Erik Ernst, AU, 2011

Målet med denne opgave er, at du kan

- Finde informationer om resurser, tilstand og hardware gennem Linux.
- Skrive et C program, der tilgår disse information.
- Få øvelse i anvendelse af `<stdio.h>`.

I Silberschatz kapitel 1 beskrives en abstakt model af et moderne computer system, hvor operativsystemet (OS) udgør et lag mellem programmer og hardware. Man kalder derfor somme tider OS for en abstrakt maskine. Opfattet som en abstrakt maskine har OS en kompleks tilstand, som blandt andet omfatter information om de resurser (herunder processoren, hukommelsen, og diverse ydre enheder) som OS administrerer. I mange varianter af OS Linux (/UNIX) kan operativsystemets tilstand inspiceres og modificeres (hvis man har de krævede rettigheder) via et særligt filsystem som oftest ligger under `/proc`, og som derfor kendes som `/proc` filsystemet. Denne facilitet tilbyder brugeren en række operationer på selve OS med et interface, som svarer til et konventionelt Linux filsystem, men hvor "filerne"er en illusion, der skabes af OS kernen for at gøre det muligt for brugeren at udføre disse operationer.

Sørg for at være logget på en Linux-maskine - se forberedelserne overfor. Gå til `/proc` filsystemet med følgende kommando (hvor `bosc$` står for din kommandoshell, som jo kan have mange forskellige former):

```
bosc$ cd /proc
```

Herefter kan man inspicere indholdet af de enkelte filer, hvilket svarer til at hente information om bestemte aspekter af operativsystemets tilstand. Eksempelvis kan man hente information om CPUen ved at inspicere indholdet af filen cpuinfo:

```
bosc$ cat cpuinfo
```

Følgende er et muligt output herfra:

```
processor : 0
vendor_id : GenuineIntel
cpu family : 6
model : 15
model name : Intel(R) Xeon(R) CPU           E5320  @ 1.86GHz
stepping : 7
cpu MHz : 1861.981
cache size : 4096 KB
physical id : 0
siblings : 4
core id : 0
cpu cores : 4
apicid : 0
initial apicid : 0
fdiv_bug : no
hlt_bug : no
f00f_bug : no
coma_bug : no
fpu : yes
fpu_exception : yes
cpuid level : 10
wp : yes
flags : fpu vme de ...
bogomips : 3726.73
clflush size : 64
power management:

processor : 1
...
```

Man kan læse meget mere om `/proc` filsystemet ved at udføre kommandoen `man proc` på en Linux maskine. Følgende C-program, `observer.c`, som kan hentes fra Bloggen, er skrevet til operativsystemet Linux. Programmet bruger `/proc` filsystemet for at få adgang til CPU modellen som operativsystemet afvikles på, til versionen af operativsystemets kerne, samt til tiden der er gået siden maskinen sidst blev genstartet.

```
/*

   Observation af kernetilstand

 */

/* API for standard input/output */
#include <stdio.h>

/* Symbolic constants */
#define DAYSEC  86400   /* seconds per day   */
#define HOURSEC 3600    /* seconds per hour  */
```

```
#define MINSEC  60      /* seconds per minut */

#define MAXBUF 512     /* lenght of largest string required to read */

/* ----  CPU model ---- */
void cpumodel(void)
{
  FILE *cpuinfofile;

  char cpumodel[MAXBUF];
  char line[MAXBUF];

  cpuinfofile = fopen ("/proc/cpuinfo","r");

  while (fgets(line,MAXBUF,cpuinfofile))
    {
      if (sscanf(line,"model name : %s",cpumodel))
      printf("CPU model: %s\n",cpumodel);
    }

  fclose(cpuinfofile);
}

/* ---- kernel version of the operating system ---- */
void kernelversion(void)
{
  FILE *versionfile;

  char kernelversion[MAXBUF];
  char line[MAXBUF];

  /* Kernel version */
  versionfile = fopen("/proc/version","r");

  fgets(line,MAXBUF,versionfile);

  if (sscanf(line,"Linux version %s",kernelversion))
    printf("Kerne version: %s\n",kernelversion);

  fclose(versionfile);
}

/* ---- uptime of the machine ---- */
void uptime(void)
{
  FILE *cpuinfofile, *versionfile, *uptimefile;

  int uptime;
```

```
  char line[MAXBUF];

  uptimefile = fopen("/proc/uptime","r");

  fgets(line,MAXBUF,uptimefile);

  fclose(uptimefile);
  if (sscanf(line,"%d.",&uptime))
    {
      int left,days,hours,minutes,secs;

      days = uptime / DAYSEC;
      left = uptime - (days * DAYSEC);

      hours = left / HOURSEC;
      left = left - (hours * HOURSEC);

      minutes = left / MINSEC;
      secs = left - minutes * MINSEC;

      printf("Uptime: %2d:%2d:%2d:%2d\n",days,hours,minutes,secs);
    }
}

/* ---- main program ---- */
int main (void)
{

  cpumodel();
  kernelversion();
  uptime();

  return 0;
}
```

3.1) Oversæt programmet og kør det. Redegør for hvorledes C-programmet fungerer. Fokuser specielt på brugen af systemkald og biblioteksrutiner, og brug resurserne (fx. opslag i man) til at belyse følgende problem: Typisk vil udskriften fra den oversatte version af `observer.c` angive "CPU model"ved en kortere streng end den som kan ses i udskriften fra `cat /proc/cpuinfo`; indse hvorfor dette er tilfældet, og prøv at få programmet til at udskrive hele teksten hørende til "CPU model".

Under `/proc` filsystemet på Linux finder man også filen `stat` som indeholder statistik om kernen og operativsystemet. Denne fil indeholder blandt andet linier på følgende form:

```
cpu  9777628 482 10390502 1451167459 1972872 41857 295863 0 0
cpu0 688403 3 475025 182603623 317730 396 3106 0 0
...
```

```
ctxt 1036176425
btime 1343982384
processes 13952327
procs_running 1
procs_blocked 0
```

Det første tal ud for teksten `cpu` specificerer hvor lang tid (målt i 1/100 sek.) systemet har været i user mode. Det tredje tal specificerer hvor lang tid systemet har været i kernel mode, og det fjerde tal specificerer den tid hvor systemet har været ubenyttet (idle). Tallet udfor teksten `ctxt` specificerer hvor mange 'context switches' systemet har udført, og tallet udfor teksten processes angiver antallet af processer der har været skabt, begge siden systemet blev genstartet. Bemærk at Linuxkernen har flere linier samt flere tal i visse linier (som vist ovenfor), men disse bruges ikke her.

3.2) Udvid C-programmet `observer.c` fra opg. 3.1 til også at give information om:

a. Den tid CPU'en har brugt i henholdsvis user mode og kernel mode.

b. Antallet af context switches.

c. Antallet af processer, der har været skabt.