IT-Universitetet i København 11. september 2014

**Operativsystemer og C**

# Ugeseddel 3 (uge 37)

---

## Forelæsning: Fredag d. 12. september kl. 10.00 - 11.50 i Aud 3.

### Emner

Processer. Procestilstande, PCB. Schedulering af processer. `fork()`, `exec()`, `wait()`, `exit()`. Kommunikation mellem processer. Delt hukommelse. Message-passing. Client-Server kommunikation, sockets og pipes. Pointers og adresser i C.

### Litteratur og andet kursusmateriale

Silberschatz, kapitel 3, afsnit 3.1-3.5 (ikke 3.5.2 og 3.5.3), 3.6 (ikke 3.6.2).

(Kernighan&Ritchie, afsnit 5.1-5.9 [Pointers og adresser], 6.1. [Structs]).

---

## Øvelser: Fredag d. 12. september kl. 12.00 - 13.50 i 4A16.

### Forberedelser (lav dette før øvelserne går i gang)

*** Læs hele opgaveteksten igennem så du ved hvad du skal til øvelserne. ***

Til øvelserne i denne uge skal du bruge

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

Obligatorisk opgave 1 omhandler implementation af en kommando-linie shell inspireret af en helt simpel kerne af Unix shell'en. Nedenstående tre opgaver om håndtering af processer kan bruges som byggesten i den obligatoriske opgave.

For let at kunne finde oplysninger om diverse systemkald, så som `fork`, `exec` og `wait`, kan du benytte `man` siderne. Du skal blot installere mapages for udviklere:

```
apt-get install manpages-dev
```

Til den obligatoriske opgave skal anvendes biblioteket `libreadline`:

```
apt-get install libreadline-gplv2-dev
```

## Opgave 1. Skabelse af proces i forgrunden og baggrunden ($\frac{1}{2}$ time)

Målet med denne opgave er, at du kan lave et C program, der kan

- Skabe en barneproces, som kører samtidigt med forældreprocessen.
- Skabe en barneproces i forgrunden og baggrunden.

I denne opgave skal der implementeres to C funktioner `foregroundcmd` og `backgroundcmd` til at starte programmer med. Filerne `forback.c` og `forback.h` indeholder en ufuldstændig version af de to funktioner, der kan benyttes som udgangspunkt.

Funktionen `foregroundcmd` skal skabe en ny proces, som udfører programmet filename med argumenterne argv. Funktionen skal herefter vente på at den skabte proces terminerer. Hint: anvend f.eks. systemkaldene `fork`, `execvp` samt `waitpid`. Man kan læse mere om de tre systemkald ved brug af man-siderne for disse systemkald. Giv argumenter for valget af parametre til systemkaldene.

Funktionen `backgroundcmd` skal skabe en ny proces, som udfører programmet filename med argumenterne angivet i argv. Funktionen skal ikke vente på at den skabte proces terminerer. Giv igen argumenter for valget af parametre til systemkaldene.

Lav et passende C program `testforback.c` som tester at funktionerne `foregroundcmd` og `backgroundcmd` fungerer efter hensigten.

## Opgave 2. Redirection af stdin og stdout (1 time)

I denne opgave skal der implementeres to C funktioner `redirect_stdincmd` og `redirect_stdoutcmd` til at starte programmer og lave "redirection"af deres standard input og standard output. Filerne `redirect.c` og `redirect.h` indeholder en ufuldstændig version af de to funktioner, der kan benyttes som udgangspunkt.

Funktionen `redirect_stdincmd` skal skabe en ny proces som udfører programmet filename med argumenterne argv, og lave redirection af programmets standard input til filen infilename. Funktionen skal herefter vente på at den skabte proces terminerer. Tilsvarende skal funktionen `redirect_stdoutcmd` skabe en ny proces som udfører programmet filename med argumenterne argv, og lave redirection af programmets standard output til filen outfilename. Hint: betragt følgende C kode fragment:

```
/* manipulate the file descriptor of the child process */
int fid = open(infilename, O_RDONLY);

/* replace stdin of the child process with fid */
close(0); /* 0 = stdin */
dup(fid);

/* close fid */
close(fid);
```

og brug man-siderne for `open`, `close` og `dup` til at lære hvorfor dette resulterer i en redirection af barneprocessens standard input.

Lav et passende C program `testredirect.c` som tester at funktionerne `redirect_stdincmd` og `redirect_stdoutcmd` fungerer efter hensigten.

## Opgave 3. Pipes mellem processer (1 time)

I denne opgave skal der implementeres en C funktion `pipecmd` til at starte to programmer og kommunikere imellem dem ved brug af en pipe. Filen `pipe.c` indeholder en ufuldstændig version af funktionen, der kan benyttes som udgangspunkt.

Funktionen `pipecmd` skal skabe to nye processer med en pipe imellem sig, som udfører programmerne filename1 med argumenterne argv1 og filename2 med argumenterne argv2. Det første programs standard input skal lytte til pipens læse-ende mens det andet programs standard output skal redirectes til pipens skrive-ende. Funktionen skal herefter vente på at de skabte processer terminerer.

Lav et passende C program `testpipe.c` som tester at funktionen `pipecmd` fungerer efter hensigten.

## Obligatorisk opgave 1

Gå igang med obligatorisk opgave 1 beskrevet i OO1E2014.pdf.