Oblig 4

Oppgave 1

---

a)

```
fredriem@itstud:~/dev/oblig4$ ./oppgave_1
Jeg er prosess 784591, min forelder er 780296
Jeg er prosess 784592, min forelder er 784591
Jeg er prosess 784593, min forelder er 784591
Jeg er prosess 784594, min forelder er 784592
```

Utskrift fra C program

---

b)

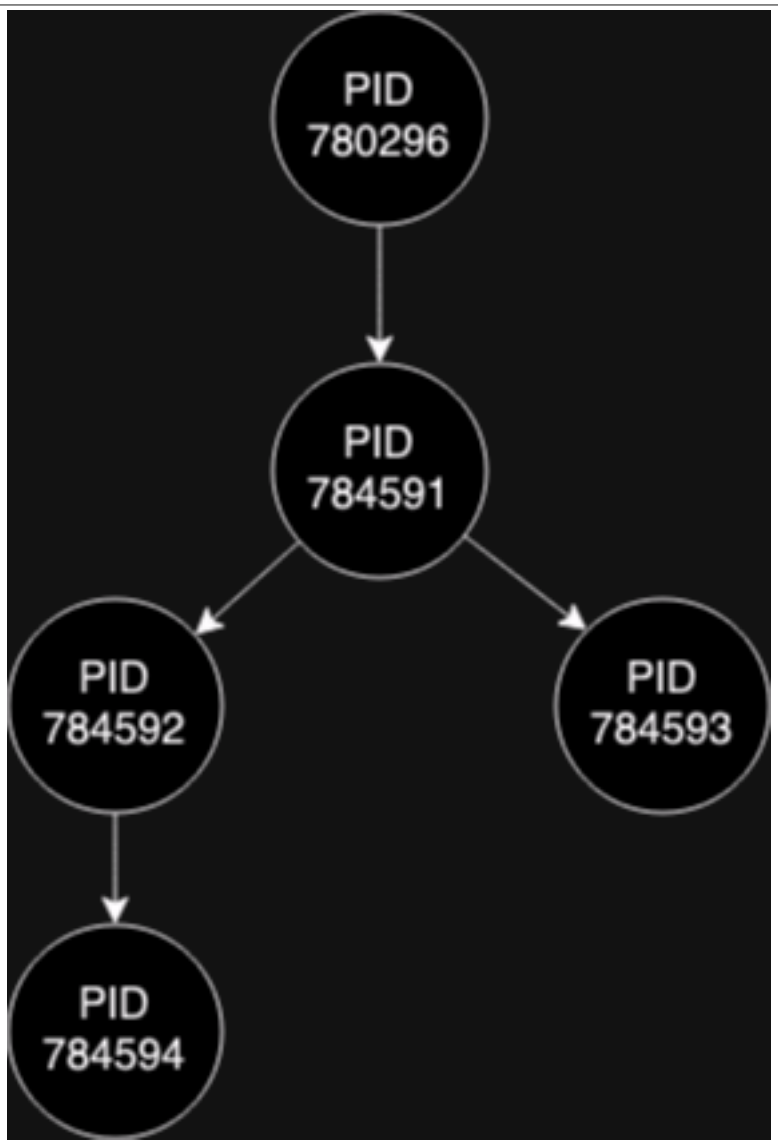

Diagram av prosess-treet

c)

Det blir opprettet 3 prosesser selv om det bare er 2 kall på fork() fordi når den første barneprosessen (child) blir opprettet med fork(), så kjører den parallellt med forelderprosessen. Så child2 = fork(); blir kjørt av både forelderprosessen og den første barneprosessen.

d)

```
fredriem@itstud:~/dev/oblig4$ ./oppgave_1d > output.txt
fredriem@itstud:~/dev/oblig4$ cat output.txt
Jeg er prosess 786643, min forelder er 780296
Jeg er prosess 786645, min forelder er 1
Jeg er prosess 786644, min forelder er 1
Jeg er prosess 786646, min forelder er 1
```

Siden barneprosessene sover i 1 sekund, så blir forelderprosessen ferdig og blir terminert før barneprosessene rekker å bli ferdig. Det som skjer da er at barneprosessene blir barnene til systemd, som har PID nr 1.

Oppgave 2

a)

```
fredriem@itstud:~/dev/oblig4$ ./oppgave_2
Jeg er barneprosessen med PID 787312
Jeg er forelderprosessen med PID 787311
Barneprosessen 787312 har terminert med returstatus lik 42
```

Før wait( ) ble tatt vekk

```
fredriem@itstud:~/dev/oblig4$ ./oppgave_2
Jeg er forelderprosessen med PID 787469
Barneprosessen 787470 har terminert med returstatus lik 0
Jeg er barneprosessen med PID 787470
```

Etter wait( ) ble tatt vekk

b)

Det første forskjellen man ser er at når wait( ) eksisterte i koden, så skriver barneprosessen først ut til terminalen. Det er fordi forelderen venter med å kjøre sin kode. Når wait( ) ble tatt vekk, så printer forelderprosessen først ut til terminalen.

Den andre forskjellen er returstatusen. I programmet med wait( ) så er returstatusen 42, men når wait( ) ble tatt vekk, så ble returstatusen 0. Den blir 42 når wait( ) er inkludert siden det er det barneprosessen returnerer.

Men når wait( ) ikke eksisterer i koden så rekker barneprosessen aldri å bli ferdig så “int status = 0” blir ikke endret etter initialisering. Så i dette tilfellet så blir utskriften misvisende, siden barneprosessen har egentlig ingen returstatus siden den blir aldri ferdig. Utskriften viser at returstatusen er lik 0 bare fordi statusen aldri blir oppdatert etter initialisering.

Oppgave 3

a)

```
fredrien@itstud:~/dev/oblig4$ ./oppgave_3 &
[1] 788761
fredrien@itstud:~/dev/oblig4$ Foreldreprosessen med PID 788761 starter en evig løkke
Barneprosessen med PID 788762 avslutter
ps -l
F S   UID     PID    PPID  C PRI  NI ADDR SZ WCHAN  TTY          TIME CMD
0 S  3352  780296  780295  0  80   0 -  2835 -      pts/5    00:00:00 bash
0 S  3352  788761  780296  0  80   0 -   591 -      pts/5    00:00:00 oppgave_3
1 Z  3352  788762  788761  0  80   0 -     0 -      pts/5    00:00:00 oppgave_ <defunct>
0 R  3352  788789  780296  0  80   0 -  2420 -      pts/5    00:00:00 ps
```

b)

```
fredrien@itstud:~/dev/oblig4$ kill 788761
fredrien@itstud:~/dev/oblig4$ ps -l
F S   UID     PID    PPID  C PRI  NI ADDR SZ WCHAN  TTY          TIME CMD
0 S  3352  780296  780295  0  80   0 -  2835 -      pts/5    00:00:00 bash
0 R  3352  789450  780296  0  80   0 -  2420 -      pts/5    00:00:00 ps
[1]+  Terminated              ./oppgave_3
fredrien@itstud:~/dev/oblig4$
```

Ved å “drepe” forelderprosessen så blir zombie prosessen også ryddet vekk av systemd. Jeg prøvde først å drepe zombien med “kill PID”, men lærte da at en zombie prosess er allerede “død” og den kan ikke drepes direkte. Eneste måten jeg fant for å fjerne den var ved å fjerne dens forelder prosess.