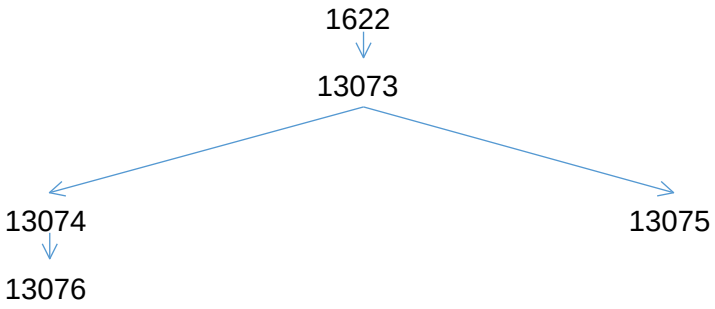
Mathias Bratz-Queseth
2023-09-22

a

```
[halambique@bluefish lab_04]$ ./oppgave_1_1
Jeg er prosess 13073, min forelder er 1622
Jeg er prosess 13074, min forelder er 13073
Jeg er prosess 13075, min forelder er 13073
Jeg er prosess 13076, min forelder er 13074
```

b



c

Programmet starter med pid **13073**, som er et barn av **1622**, bash-terminalen jeg kjører programmet fra**. 13073** forkes deretter til **13074** når *child1 = fork()* blir utført.

Programmet kjører nå to prosesser: **13073** og **13074.** Når *child2 = fork()* utføres, blir begge disse prosessene også gaflet, til prosessene **13076** (barn av **13074**) og **13075** (barn av **13073**).

d

Jeg er prosess 13545, min forelder er 1622
Jeg er prosess 13547, min forelder er 741
Jeg er prosess 13546, min forelder er 741
Jeg er prosess 13548, min forelder er 741

Fordi kodelinjene med if-statements får foreldreprosessene til å sove i ett sekund, vil foreldreprosessen deres igjen (**741**)overta barneprosessene **13545, 13547, 13546, 13548.**

a

```
[halambique@bluefish oppgave_2]$ ./oppgave_2
Jeg er barneprosessen med PID 10512
Jeg er forelderprosessen med PID 10511
Barneprosessen 10512 har terminert med returstatus lik 42
```

b

```
[halambique@bluefish oppgave_2]$ ./oppgave_2
Jeg er forelderprosessen med PID 10619
Barneprosessen 10620 har terminert med returstatus lik 0
Jeg er barneprosessen med PID 10620
```

Når vi fjerner wait-kallet tillates det at foreldreprosessen eksekveres uten å vente på at barneprosessen avsluttes. Dette fører til uforutsigbar rekkefølge i eksekveringen, og forklarer hvorfor ikke følgende kodelinje fører til siste utskrift i terminalen:

```
printf("Barneprosessen %d har terminert med returstatus lik %d\n",
      child, WEXITSTATUS(status));
```

a

```
[halambique@bluefish oppgave_3]$ Forelderprosessen med PID 11122 starter en evig løkke
Barneprosessen med PID 11123 avslutter
ps -l
F S   UID     PID    PPID  C PRI  NI ADDR SZ WCHAN  TTY          TIME CMD
0 S  1000    1549    1535  0  80   0 -  2075 do_wai pts/1    00:00:00 bash
0 S  1000   11122    1549  0  80   0 -   723 hrtime pts/1    00:00:00 oppgave_3
1 Z  1000   11123   11122  0  80   0 -     0 -      pts/1    00:00:00 oppgave_3
4 R  1000   11125    1549  0  80   0 -  2895 -      pts/1    00:00:00 ps
```

b

```
pkill -TERM -g 11122
```

Denne kommandoen vil drepe foreldreprosessen (som er i en evig loop) og dermed også barneprosessen. Et alternativ, hvis man vet navnet på kommandoen som initierte prosessen ved bruk av en spesiell variabel for å henvise til riktig prosess i tabellen:

```
kill $(pgrep oppgave_3)
```