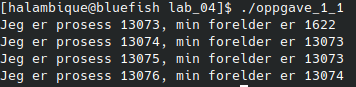
Oppgave 1

a



b

|  |  |
| --- | --- |
| 1622 | |
| 13073 | |
| 13074 | 13075 |
| 13076 |  |

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

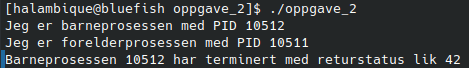
Jeg er prosess 13546, min forelder er 741

Jeg er prosess 13548, min forelder er 741

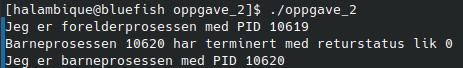
Fordi kodelinjene med if-statements får foreldreprosessene til å sove i ett sekund, vil foreldreprosessen deres igjen (**741**)overta barneprosessene **13545, 13547, 13546, 13548.**

Oppgave 2

a



b

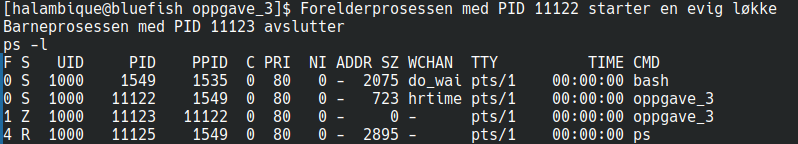


Når vi fjerner wait-kallet tillates det at foreldreprosessen eksekveres uten å vente på at barneprosessen avsluttes. Dette fører til uforutsigbar rekkefølge i eksekveringen, og forklarer hvorfor ikke følgende kodelinje fører til siste utskrift i terminalen:

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

Oppgave 3

a



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*)