Oppgave 1:

Deloppgave A: Kompiler og kjør programmet Et bilde som inneholder tekst, Font, skjermbilde

Automatisk generert beskrivelse

Deloppgave B: Figur av prosestreet

Et bilde som inneholder tekst, skjermbilde, Font, nummer

Automatisk generert beskrivelse

Deloppgave C: Kort forklaring

Selv om *fork* kun kalles to ganger, opprettes det tre nye elementer. Første gangen *fork* blir brukt, opprettes det én ny prosess, nemlig det første barnet (Child 1). Andre gangen *fork* blir brukt, blir det opprettet enda et nytt barneelement (Child 2), men også et barnebarn av første *fork* (Grandchild)*.* Dette er fordi sisten gangen vi kaller på *fork* kjøres den både av den opprinnelige prosessen og av barnet som ble opprettet ved første *fork.*

Deloppgave D:



Et bilde som inneholder tekst, skjermbilde, Font

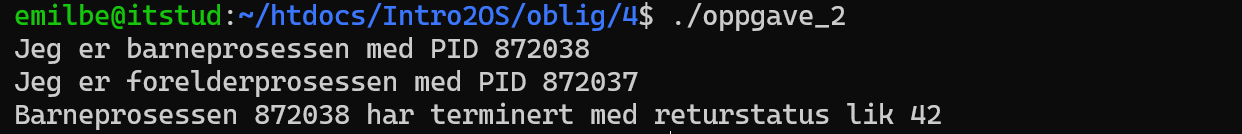
Automatisk generert beskrivelse

Over ser du at jeg har redirigert utskriften av c-programmet til en .txt fil som jeg deretter har åpnet med *cat*.

Vi ser på utskriften at mange av elementene har fått forelder PID 1. Grunnen til dette er at forelderen til noen prosesser avsluttes tidlig, og barneprosessene blir adoptert av init (stamfar), som har PID = 1. Kort sagt blir egentlig disse barneprosessene foreldreløse, og systemet gir derfor foreldrerettighetene til «systemd» som adopterer disse prosessene. Hvis man vil unngå at dette skjer, må man «vente på tur», som vi si av hvert barneelement må få fullføre, før forelderen avslutter, eksempelvis ved å bruke «wait()».

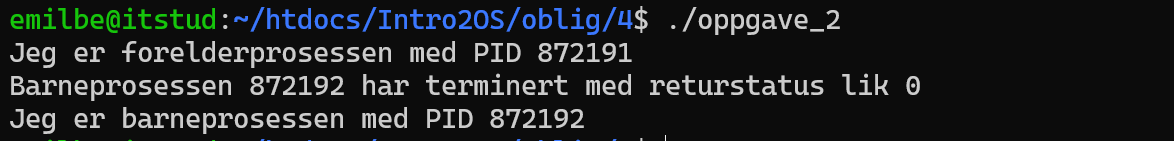
Oppgave 2

Deloppgave A: Utskrift



Her vil barneprosessen kjøre først, sove i ett sekund, og avslutter deretter med returstatus 42. Foreldreprosesen venter til barneprosesen er ferdig og skriver deretter ut returstatusen.

Deloppgave B: Utskrift uten wait()



Forskjellen mellom utskriften i Oppgave A og B er flere. Først og fremst vil forelderprosessen kjøre først, ettersom den ikke må vente på at barneelementet blir ferdig, grunnet fjerningen av wait() og ettersom barneprosessen har en soveperiode på ett sekund. I tillegg vil den ikke inkludere returstatus 42, ettersom forelderprosessen kjører før barneprosessen, som vil si at barneprosessen ikke har rukket å returnere 42. Programmet kjører derfor nå i «feil» rekkefølge, med tanke på resultatene vi egentlig vil ha.

Oppgave 3

Deloppgave a: ps -l

Et bilde som inneholder tekst, skjermbilde, Font

Automatisk generert beskrivelse

Deloppgave B: Bli kvitt zombieprosessen

For å bli kvitt zombieprosessen kan man «drepe» forelderelementet til zombien. Dette kan gjøres ved å skrive: eventuelt . Denne virker som “sure kill” og sikrer at prosessen faktisk blir terminert.

Et bilde som inneholder tekst, skjermbilde, Font

Automatisk generert beskrivelse