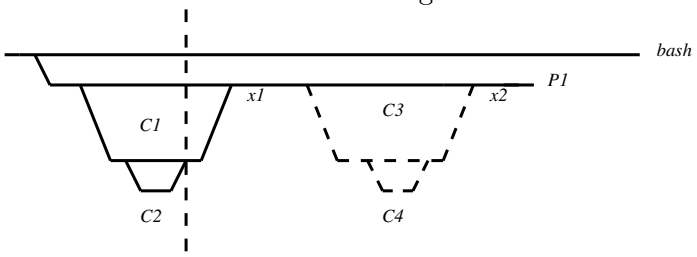
Inför datortentan I

Syftet med detta dokument (och efterföljande dokument) är att presentera exempel på problemställningar som kan komma på datortentan. Vi ska studera lösningar till de problemställningar som presenteras här på seminariet som hålls nästa vecka. Det är därför högst lämpligt att gä seriösa försök till lösningar på dessa problem.

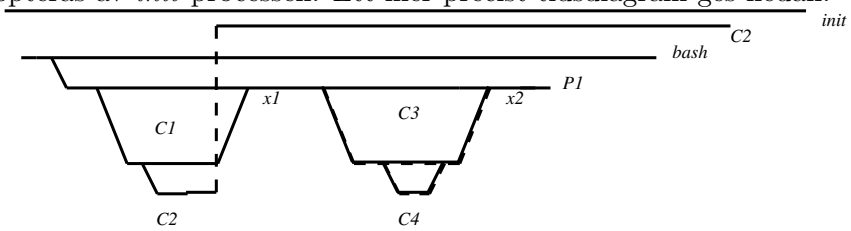
Exempel 1

Vi studerade ett exempel på ett kursmöte på ett program som skapade ett antal processer som kunde beskrivas med nedanstående tidsdiagram:



Allstå en föräldraprocess, benämnd *P1* som i sin tur är barn till en skalprocess, benämnd *bash*. Processen *P1* skapar barnprocesserna benämnda *C1* och *C3* som i sin tur skapar sina respektive barnprocesser benämnda *C2* och *C4* som skulle kunna kallas barnbarn och då skulle även processerna *C2* och *C4* kunna kallas "kusiner".

Tidsdiagrammet är utritat i en situation där inte ännu processerna *C2* och *C4* är skapade. Vidare är inte tidsdiagrammet riktigt överensstämmande med hur det ska fungera i framtiden, vi vill nämligen att *C2* ska adopteras av *init*-processen. Ett mer precist tidsdiagram ges nedan.



Här illustreras vad som händer när *C2* dör, dess barnprocess adopteras av *init*. Dessutom ska programmet (som består av alla dessa processer tillsammans) göra utskrifter av alla processer som visas med kommandot `ps`, föräldraprocessen ger kommandot

```
system("ps -e -o pid,ppid,pgid,sess,comm");
```

vid de två tidpunkterna markerade med *x1* respektive *x2* i figuren ovan. Då ges en presentation av situationen i form av resultatet från det kommandot. Vidare ska även barnet som adopteras av *init* kommunicera via en pipe som går från *C4* till *C2* där *C4* ska skicka sin processid till *C2*. Ena kusinen ska alltså berätta vilket processid den har för den andra. Sedan ska *C2* skriva ut sitt eget processid och kusinens.

Detta program finns på kurswebben och utgör ett exempel på en problemställning av en typ som skulle kunna vara intressant att ta in på datortentan.

Nedan följer ett antal övningar som ni kan göra för att träna på att hantera dessa typer av problem. Vi skapar fyra varianter av det ovanstående programmet:

1. Byt roller på kusinerna, låt *C4* vara den som bli adopterad av *init* istället.
2. Låt *P1* skapa ytterligare ett barn som går in mitt emellan *C4* och *C2* och som dubblerar det som skickas, tag koden frå processen som heter `double` och som gavs på kursmötet om relationer mellan processer. *Ledning:* för att anropa den krävs användning av `execlp` och anropet blir

   ```
   execlp("./double","./double",NULL);
   ```

   och då måste programkoden hörande till `double` ligga i arbetskatalogen för *P1*.
3. Skriv om *C2* så att den kan få flera heltal skickade till sig som användaren skriver in, skapa ett snyggt avslut liknande det som beskrivits på kommunikationsövningen, där *C2* läser från pipen med ett anrop av typen `while(read(...))`.
4. Gör samma variant som övning 2 strunta i att skapa en barnprocess som blir `double`, skriv bara om processen *C3* så att den tar processid:t frå *C4* och skickar vidare till *C2*.

I alla övningarna, behåll anropet till `system("ps ...")`.