
## Första seminariet i Operativsystem, HI1025:TEN1

Syftet med det här och nästa seminarium är att arbeta tillsammans för att ni ska klara tentorna. Det här första seminariet kommer att ha fokus på den skriftliga tentan, men nästa seminarium kommer har mer fokus på datortentan. Båda seminarierna behöver utvärderas så att vi vet vad som är bra och vad som är dåligt.

### Tentorna

Kursen har som ni vet två tentor, en skriftlig tenta och en datortenta. Den skriftliga tentan är en vanlig salsskrivning med betyg A-F och datortentan har endast betygen P och F.

**Den skriftliga tentan.** På den skriftliga tentan ges följande instruktioner (48 poäng är bara ett exempel):

”Tentamen innehåller 10 frågor/uppgifter med totalt 48 poäng. För lägsta godkända betyg (E) krävs ungefär 22-23 poäng. För att få komplettera (Fx) krävs ungefär 20-21 poäng.
**Viktigt:** När du svarar på en fråga ska det i allmänhet framgå varför du vet svaret frågan, det uppnås enklast genom att du sätter in termer och begrepp som du hanterar i deras sammanhang på ett korrekt sätt. Men det är också viktigt att inte bli för mångordig, i uppgifter med 1 poäng krävs ett kort svar, i uppgifter med mer poäng behöver svaret utvecklas mer. **OBS: Svara *inte* i tentan, *bara* på svarspapper.**

Det vi ska studera i detta seminarium är hur vi ska skapa formuleringar på tentan som uppfyller detta och vi ska också få klart för oss hur vi ska studera för att kunna svara på tentan på det sätt som krävs. Innan vi dyker in i detta poängterar vi att dessa instruktioner har sin grund i målen med utbildningen, det står i utbildningsmålen följande:

för ingenjörsexamen ska studenten ”visa förmåga att med helhetssyn självständigt och kreativt identifiera, formulera och hantera frågeställningar och analysera och utvärdera olika tekniska lösningar ... ”.

Den skriftliga tentamen tränar alltså denna förmåga, särskilt viktigt är ordet helhetssyn och det är med helheten i beaktning som tentorna kommer att rättas: texten som studenterna skriver ska vara sammanhängande och kunniga. Vi ska idag trända detta och se på en student som misslyckats med att formulera sig och en student som lyckats. De svar som jag presenterar utgör verkliga exempel.

### Exempel på uppgifter

(1) (Fråga nummer 1 på förra årets första tenta.) Förklara vad *time-sharing* är och hur det går till. I din redogörelse ska nyckelorden *process*, *CPU*, *context switch* ingå och vara insatta i sammanhanget på ett korrekt sätt. Det ska klart framgå hur *time-sharing* virtualiserar CPU:n. **(4p)**

Att nyckelorden anges är ett stöd till studenten som ska svar på frågan. Det ges total 4 poäng, ett för varje nyckelord som är korrekt använt och ett ett sista poäng för hur det framgår att virtualisering realiseras.

(På själva seminariet kommer det ges tid att ta fram förslag till svar på frågorna, vi kommer att diskutera om svaren är fullständiga där och då).

(2) (Fråga nummer 3 på förra årets tenta.) Förklara följande systemanrop vad de gör, varför de är nödvändiga. Förklara hur de samverkar **(1p)**.
   (a) `fork()` **(2p)**
   (b) `exec()` (principen bakom `execve()`/`execlp()`/`execl()` etc.) **(2p)**
   (c) `wait()` **(2p)**

Återigen ser vi hur frågan är uppdelad i delar, 2 poäng för varje korrekt svar och varje svar ska bestå av två delar: den första vad `fork` är, den andra hur den samverkar med de andra systemanropen.

(På själva seminariet kommer det ges tid att ta fram förslag till svar på frågorna, vi kommer att diskutera om svaren är fullständiga där och då).

(3) (Fråga 4 på samma tenta.) Förklara skillnaden mellan ett systemanrop och ett anrop av en vanlig funktion, vad menas med att man måste kontrollera returvärdet från ett systemanrop och varför måste vi det? Ge ett konkret exempel på hur du gjort då du arbetat med övningar och laborationer där detta förekommer och motivera just varför du kontrollerat returvärdet av det systemanrop du exemplifierar. **(4p)**

(4) (Fråga 5 på samma tenta.) Förklara vad en DMA är och hur den ökar användningen av CPU:n i ett operativsystem jämfört med om systemet inte skulle ha en DMA. **(2p)**

(5) (Fråga 6 på samma tenta.) Förklara skillnaden mellan hårda och symboliska (mjuka) länkar och ange deras användningsområden. Se till att det användningsområde du anger för symboliska länkar inte fungerar med hårda länkar. **(4p)**

(6) (Fråga 7 på samma tenta.) Du arbetar som systemadministratör på ett företag. Det system du arbetar med har olika partitioner monterade i en traditionell *UNIX*-filhierarki. (`/dev/sdaX`, `X=1,2,3,...`.) Katalogen `/var` är separat monterad på `/dev/sda6`. Du vill nu byta filsystemstyp till `XFS` på den partitionen men förstås inte förstöra innehållet. Hur gör du detta? Du har tillgång till en reservhårddisk (`/dev/sdb1`) för att mellanlagra data när du formaterar om. Ange i detalj, steg för steg, hur du skulle använda kommandona `mkdir`, `cp` (eller `mv`), `mkfs.xfs` (skapar ett XFS-filsystem), och liknande kommandon för att åstadkomma detta. *Ledning:* Ett av delstegen är `mkfs.xfs /dev/sda6` som ju formaterar om partionen som `/var` är monterad på men det måste göras på rätt sätt så att data inte förstörs. **(3p)**

**Den Praktiska Tentan.** Vi kommer att arbeta med samma plattform som i kursen i grundläggande programmering, ni kommer in i en Linux-miljö med tillgång till Code-Blocks och manualsidor.

Vi kommer också att tillåta hjälpmedel på praktiska tentan i form av en samling av exempelkoder. Vi upprätter detta hjälpmedel tillsammans på seminarierna i kursen. Jag förvaltar den officiella versionen av hjälpmedelssamlingen och kommer att se till att programkoden finns tillgänglig på tentakontona under tentadagen. Som examinator kommer jag också att vara den som avgör vad som kan vara rimligt att ta med i exempelsamlingen.

**Exempelsamling.** Vi utvidgar exempelsamlingen gradvis under kursen. Ni är välkomna att föreslå tillägg under kursens gång, föreslå dessa genom att skicka ett mejl till mig. Nu i början lägger jag exempelsamlingen här men så småningom ska jag lägga den på kursens webbsida synlig för alla. Ni får dock inte ta med den i pappersform på tentan, jag kommer att se till att den finns tillgänglig på datorerna vid tentadagen som sagt.

*Användbara systemanrop.*

```
fork() exit() wait() getpid() getppid() execl() execlp() pipe() open() close()
```

*Hitta en includefil (och mer information) för ett systemanrop.*

```
$ man <systemanropet>
```

*Skapa en parallell process.*

```
pid_t pid;
pid = fork();
switch(pid)
{
  case -1: fprintf(stderr,"Fork failed.\n"); exit(1);
  case 0: childcode(); exit(0); //Körs i en parallell barnprocess.
  default: parentcode1(); wait(0); //Körs parallellt med barnprocessen.
}
parentcode2(); //Körs efter barnprocessen avslutat.
```