Tentamen i Operativsystem, HI1025:TEN1 - 9 juni, 2015

Allmänna instruktioner. Tentamen innehåller 10 frågor med totalt x poäng. För lägsta godkända betyg (E) krävs ungefär $x/2$ poäng. För att få komplettera (Fx) krävs ungefär $x/2 - 2$ 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.**

Anmärkning: Av layoutskäl saknas nödvändiga `#include`-direktiv i källkoden i vissa av uppgifterna, men uppgifterna ska behandlas som om direktiven fanns där.

Uppgifter

(1) Vad är ett operativsystems uppgift? Det finns en situation idag där man använder ett minimalt operativsystem (som alltså inte gör så mycket alls), berätta om den situationen och motivera varför operativsystemet är så litet. **(2p)**

(2) Förklara kort följande begrepp:
   (a) *Broken pipe*
   (b) *Disk Striping*
   (c) *Thread Synchronization/Trådsynkronisering*
   (6p)

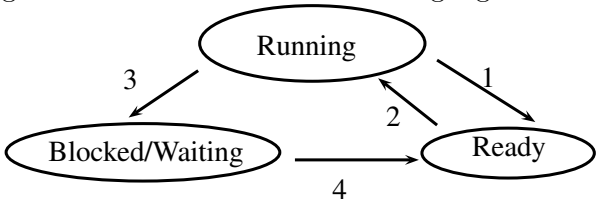
(3) Förklara vad hårda och mjuka länkar i ett UNIX-filsystem är och förklara särskilt varför man gärna föredrar att använda mjuka länkar. **(3p)**

(4) Vad är ett *processkontrollblock*, vad innehåller det och vad används det till? **(3p)**

(5) När en demon kör så vill man att demonen ska påverka operativsystemet så lite som möjligt och omvänt vill man att operativsystemet ska påverka demonen så lite som möjligt. Nämn två olika åtgärder som man vidtar vid start av en demonprocess, en som innebär att demonen påverkar sin omgivning mindre och en som gör att omgivningen påverkar demonen mindre och förklara konkret varför åtgärderna får leder till att demonen och dess omgivning påverkar varandra mindre. **(4p)**

(6) Förklara de tre begreppen *Stricly non-preemptive*, *Strictly preemptive* och *Politely preemptive* och hur de tillämpas (eller tillämpad*es*). **(3p)**

(7) Då en process finns i ett operativsystem så kan den vara i olika tillstånd. Betrakta nedanstående diagram som beskriver tillståndsövergångarna för en process:



Beskriv vad som kan utlösa var och en av de olika övergångarna och vad som då händer. **(4p)**

(8) Ange 3 saker som är unika för varje process (men samma för varje tråd i processen) och ange 3 saker som är unika för varje tråd som kör i en och samma process. **(3p)**

(9) Studera följande programkod:

```
c(){if(!fork())exit(0);else wait(0);}
cc(){if(!fork()){c();exit(0);} else wait(0);}

main(){
  if(!fork()){sleep(1);exit(0);}
  cc(); wait(0);
}
```

Rita ett tidsdiagram som tydliggör släktförhållandena mellan de processer som startas till följd av att man kör detta program. Hur många processer är maximalt igång samtidigt till följd av att detta program startas? Motivera ditt svar genom att hänvisa till ditt tidsdiagram. **(5p)**

(10) Studera följande programkod:

```
main() {
  int run=10, fd[2]; pipe(fd); srand(time(0)); char ch, x='x', o='o';
  if(!fork()) {
    close(fd[0]);
    while(run){sleep(1+rand()%6); write(fd[1],&x,1);}
    exit(0);
  }
  if(!fork()) {
    close(fd[0]);
    while(run){sleep(1+rand()%3); write(fd[1],&o,1);}
    exit(0);
  }
  close(fd[1]);
  while(run--)
  {
    read(fd[0],&ch,1);
    printf("%d %c\n",run,ch);
    sleep(1);
  }
  wait(0); wait(0);
}
```

Vi har som vi ser två barnprocesser och en föräldraprocess. Vad förväntar du dig för utseende på detta programs körning? Det finns ett allvarligt problem med detta program som gör att det aldrig kan fungera som det är tänkt, beskriv problemet och ge en övergripande beskrivning av en lösning av det problemet. **(6p)**