UMEÅ UNIVERSITET
Institutionen för datavetenskap
Ola Ågren

Lösningsförslag Datavetenskap

Kurs:	**Systemprogrammering, 5p**
Tentamensdatum:	**19 November 1997**
Hjälpmedel:	**Miniräknare**
Examinator:	**Ola Ågren (ola@cs.umu.se)**

G/3 = 30, 4 = 34.8, VG = 38, 5 = 39.6

Uppgift 1 (14p)

Skriv namn, personnummer, uppgift (max en per papper) och sidnummer på därför avsedd plats på varje sida och använd endast ena sidan av varje papper.

Svar:

Inte allt för svårt hoppas jag...

Uppgift 2 (4p)

Det finns fem sätt att avsluta en process, beskriv dessa (en poäng per sätt, men inte fler än fyra räknas).

Svar:

1. Att göra *return* från main.
2. Att anropa *exit()*.
3. Att anropa *_exit()*.
4. Att anropa *abort()*.
5. Att få en *signal* som man inte tar emot eller ignorerar.

Uppgift 3 (4p)

Inom systemprogrammering pratar man ofta om processer och hur de interagerar med varandra. Definiera alt. föklara vad en process är samt rita och beskriv det virtuella minnesutrymmet för en process.

Svar:

En process är en just nu exekverande instans av ett program. Den har sitt eget minnesutrymme som ser ut som följer:

Stack	Växer nedåt allt efter behov
Tomrum	Just nu ledigt minne
Allokerat minne/Heap	Program och systemstyrt, växer uppåt
Data	Här finns alla globala variabler
Text	Program och konstanta värden

Det finns maskiner där stacken och det allokerade minnet har bytt plats och sålunda växer åt andra hållet. Jag ger naturligtvis inte fel för ett sådant exempel.

Uppgift 4 (4p)

Nämn och beskriv fyra sätt att få processer att synkroniseras med varandra.

Svar:

signaler	Sparka på en process
semaforer	Låser ute varandra, sk mutex (mutual exclusive)
låsfiler/låsbibliotek	Mutex över filsystem (låsfiler ej i NFS-miljö)
monitorer	Funktioner som sköter mutex runt problemområden/resurser
pipor	Skicka data mellan processer med gemensam anfader för synkronisering
namngivna pipor (FIFOs)	En process kan öppna för läsning, en för skrivning, i övrigt som pipor
meddelandeköer	Man skickar meddelanden till en kö som någon annan process sedan kan hämta och läsa
delat minne	En process ändrar i minnet och alla andra ser det genast
minnesmappade filer	I princip samma sak som delat minne, men filen finns även på disk

Meddelandeköer är inte sann synkron synkronisering, men det kan man väl inte anklaga pipor eller FIFO att vara heller...

Uppgift 5 (6p)

Beskriv vad som händer när följande kommando exekveras.

```
% sed -e script < indata | tee utdata
```

Hur startas processer och program? När, och av vilken process skapas pipor? Hur hanteras omdirigering etc? Beskriv utförligt och rita gärna figurer.

Svar:

Före:

csh (run)
0:term
1:term
2:term

Efter **pipe**:

csh (run)
0:term
1:term
2:term
3:pipe, read end
4:pipe, write end

Efter 2***fork**:

csh (run)
0:term
1:term
2:term
3:pipe, read end
4:pipe, write end

csh (run)
0:term
1:term
2:term
3:pipe, read end
4:pipe, write end

csh (run)
0:term
1:term
2:term
3:pipe, read end
4:pipe, write end

Efter 2***dup**:

csh (run)
0:term
1:term
2:term
3:pipe, read end
4:pipe, write end

csh (run)
0:term
1:pipe, write end
2:term
3:pipe, read end
4:pipe, write end

csh (run)
0:pipe, read end
1:term
2:term
3:pipe, read end
4:pipe, write end

Efter 6***close**:

csh (wait)
0:term
1:term
2:term

csh (run)
0:term
1:pipe, write end
2:term

csh (run)
0:pipe, read end
1:term
2:term

Efter **open**:

csh (wait)
0:term
1:term
2:term

csh (run)
0:indata
1:pipe, write end
2:term

csh (run)
0:pipe, read end
1:term
2:term

Efter 2***exec**:

csh (wait)
0:term
1:term
2:term

sed (run)
0:indata
1:pipe, write end
2:term

tee (run)
0:pipe, read end
1:term
2:term

Uppgift 6 (2p)

Vad är en hård länk? Vad är en symbolisk länk? Beskriv någorlunda utförligt likheter och skillnader dem emellan!

Svar:

En hård länk är ett sätt att ge ytterligare ett namn på en fil så att man kommer åt den med fler än en sökväg. Om man tar bort den ena länken så kommer den andra fortfarande att komma åt den gamla filen. En hård länk måste ligga på samma partition som den ursprungliga filen.

En symbolisk länk är en fil som innehåller en sökväg till en fil eller ett bibliotek. Den utpekade filen måste inte existera, men måste ha ett giltigt namn. Filen kan dessutom ligga på en annan partition.

Uppgift 7 (6p)

Rita hur pekarstrukturen ser ut vid början av exekveringen och skriv dessutom vad som skrivs ut:

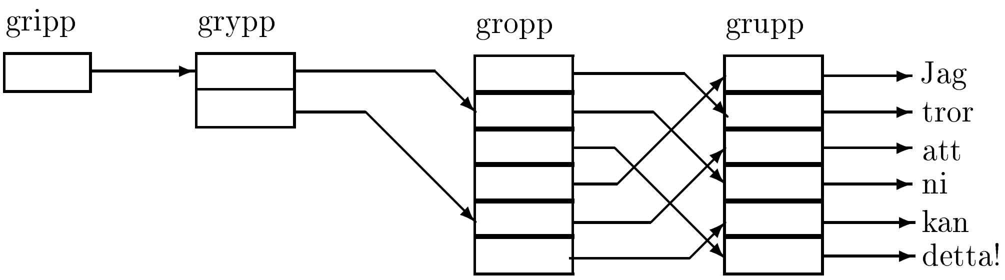
```
#include <stdio.h>
typedef char grepp[10];

grepp grupp[] = {"Jag", "tror", "att", "ni", "kan", "detta!"};
grepp *gropp[] =
    {&grupp[1], &grupp[3], &grupp[5], &grupp[0], &grupp[2], &grupp[4]};
grepp **grypp[] = {&gropp[1], &gropp[4]};
grepp ***gripp = grypp;

int main()
{
  char *grapp = (**gropp) + 2;

  (*grypp[1][1])[2] = '\0';
  (*grypp[1][-2])[5] = '\0';
  printf("%s%s", grapp++, grypp[1][1]);
  printf("%s %s", grapp, ***gripp);
  printf(" %s?\n", **(gropp + 2));
  exit(0);
}
```

Svar:




```
orkar ni detta?
```

Uppgift 8 (6p)

Ni har just fått anställning på aktiebolaget Minimal Softwares (**MS**). Eftersom det finns ett fel i programmet `wc` på en av maskinerna så blir er första uppgift att skriva programmet `mwc` som skall fungera som `wc -cl`. Eftersom ni just nu inte har någon bok med er följer (efter mansidan för `wc`) några funktionsprototyper som ni eventuellt kan ha använding av.

...

Svar:

```
#include <stdio.h>

int main(int argc, char *argv[])
{
    FILE *fp;
    int i, next, chars, lines, totalChars, totalLines;

    totalChars = totalLines = 0;
    if (argc < 2) {
        chars = lines = 0;
        while ((next = getchar()) != EOF) {
            chars++;
            if (next == '\n')
                lines++;
        }
        printf("%8d%8d\n", chars, lines);
    }
    else
        for (i = 1; i < argc; i++) {
            chars = lines = 0;
            if ((fp = fopen(argv[i], "r")) != NULL) {
                while ((next = getc(fp)) != EOF) {
                    chars++;
                    if (next == '\n')
                        lines++;
                }
                fclose(fp);
                printf("%8d%8d %s\n", chars, lines, argv[i]);
                totalChars += chars;
                totalLines += lines;
            } else
                perror(argv[i]);
        }
    if (argc > 2)
        printf("%8d%8d total\n", totalChars, totalLines);
    exit(0);
}
```