

Operativsystem ID1200/06

(ID2200/06 6hp)

Tentamen

2019-04-16 14:00-18:00

Instruktioner

- Du får, förutom skrivmateriel, endast ha med dig en egenhändigt handskrivnen A4 med anteckningar. Anteckningarna lämnas in och kan inte återanvändas.
- Svaren skall lämnas på dessa sidor, använd det utrymme som finns under varje uppgift för att skriva ner ditt svar.
- Svar skall skrivas på svenska eller engelska.
- Du skall lämna in hela denna tentamen och den handskrivna sidan med anteckningar. Inga ytterligare sidor skall lämnas in.

Betyg

Tentamen har ett antal uppgifter där några är lite svårare än andra. De svårare uppgifterna är markerade med en stjärna, *poäng**, och ger poäng för de högre betygen. Vi delar alltså upp tentamen i grundpoäng och högre poäng. Se först och främst till att klara grundpoängen innan du ger dig i kast med de högre poängen.

För frågor med flera delfrågor ges normalt 2p för alla rätt och 1p för ett fel.

Notera att det av de 12 grundpoängen räknas bara som högst 11 och, att högre poäng inte kompenserar för avsaknad av grundpoäng. Gränserna för betyg är som följer:

- Fx: 6 grundpoäng
- E: 7 grundpoäng
- D: 8 grundpoäng
- C: 10 grundpoäng
- B: 11 grundpoäng och 5 högre poäng
- A: 11 grundpoäng och 8 högre poäng

Gränserna kan komma att justeras nedåt men inte uppåt.

Namn: _____ Persnr: _____

1 Processer

1.1 stack eller heap [2 poäng]

Vad gör proceduren nedan och var skall **gurka** allokeras, på stacken eller *heapsen*? Varför? Skriv färdigt koden så att gurka blir allokerat utrymme.

```
int *tomat(int *a, int *b) {  
    // allocate room for gurka  
  
    *gurka = *a + *b;  
    return gurka;  
}
```

Namn: _____ Persnr: _____

1.2 fork() [2 poäng]

Vad skrivs ut när vi kör programmet nedan, vilka alternativ finns och varför får vi detta resultat?

```
int global = 17;

int main() {
    int pid = fork();
    if(pid == 0) {
        global++;
    } else {
        global++;
        wait(NULL);
        printf("global = %d \n", global);
    }
    return 0;
}
```

Namn: _____ Persnr: _____

1.3 `__sync_val_compare_and_swap()` [2 poäng*]

Vi kan implementera ett spinn-lås i GCC enligt nedan. Låset implementeras med en maskininstruktion som atomärt jämför värdet i en minnesposition och om den uppfyller det krav vi har, ersätter den med ett nytt värde. I implementationen nedan så representerar vi ett öppet lås med värdet 0; om låset är öppet så skriver vi en 1 i den angivna positionen och returnerar 0, i annat fall returnerar vi det funna värdet (som då torde vara 1).

Antag att vi använder låset för att synkronisera två trådar på en maskin med en kärna; vad är då nackdelen som vi kommer ha? Hur skulle vi kunna åtgärda den nackdelen?

```
int try(volatile int *mutex) {
    return __sync_val_compare_and_swap(mutex, 0, 1);
}

void lock(volatile int *mutex) {
    while(try(mutex) != 0) { }
}

void release(volatile int *mutex) {
    *mutex = 0;
}
```

Namn: _____ Persnr: _____

2 Kommunikation

2.1 från den ene till ... [2 poäng]

Antag att vi har två program, `ones` och `add2`, implementerade enligt nedan. Anropet till `scanf("%d", &in)` läser från `stdin` och parsar ett tal som sedan skrivs till `&in`. Proceduren returnerar antingen 1, om den kunde läsa ett tal, eller `EOF`. Anropen till `printf()` kommer skriva ut talen på `stdout`.

```
/* ones.c */
#include <stdlib.h>
#include <stdio.h>
```

```
int main() {
    for(int n = 5; n > 0; n--) {
        printf("%d\n", n);
    }
    return 0;
}
```

```
/* add2.c */
#include <stdlib.h>
#include <stdio.h>

int main() {
    int in;
    int result = scanf("%d", &in);

    while(result != EOF) {
        printf("%d\n", in+2);
        result = scanf("%d", &in);
    }

    return 0;
}
```

Till ditt förfogande har du en Linux-dator med alla tänkbara program. Hur skulle du på enklast möjliga sätt få utskriften från det ena programmet, `ones`, att läsas av det andra, `add2`.

Namn: _____ Persnr: _____

2.2 Delat minne [2 poäng*]

Vi kan låta två processer dela minne genom att låta båda processerna minnesmappa en fil mha `mmap()`. Detta minne blir då synligt för båda processerna och de kan dela det, nästan precis på samma sätt som två trådar delar heap.

I exemplet nedan så finns koden för att mappa en delad fil som sen kan användas av flera processer. Vi har också ett utdrag ur manualen för `mmap()`.

```
int fd = open("shared", O_CREAT | O_RDWR, S_IRUSR|S_IWUSR);

// make sure the file is 4K byte
lseek(fd, 4096, SEEK_SET);
write(fd, "A", 1);

char *area = (char*)mmap(NULL, 4096, PROT_READ | PROT_WRITE, MAP_SHARED, fd, 0);
:
:

void *mmap(void *addr, size_t length, int prot, int flags, int fd, off_t offset);
```

DESCRIPTION

`mmap()` creates a new mapping in the virtual address space of the calling process. The starting address for the new mapping is specified in `addr`. The `length` argument specifies the length of the mapping (which must be greater than 0).

If `addr` is `NULL`, then the kernel chooses the (page-aligned) address at which to create the mapping; this is the most portable method of creating a new mapping. If `addr` is not `NULL`, then the kernel takes it as a hint about where to place the mapping; on Linux, the mapping will be created at a nearby page boundary. The address of the new mapping is returned as the result of the call.

:
:

Vad skulle hända om vi vill dela länkade datastrukturer, vilket problem skulle vi ha och hur skulle man kunna hantera det?

Namn: _____ Persnr: _____

3 Schemaläggning

3.1 Bonnie Tylor [2 poäng]

Antag att vi har en schemaläggare som använder shortest job first. Vi har fyra jobb som nedan anges med $\langle \text{anländer vid, exekveringstid} \rangle$ i ms. Rita upp ett tidsdiagram över exekveringen och ange omloppstiden för vart och ett av jobben.

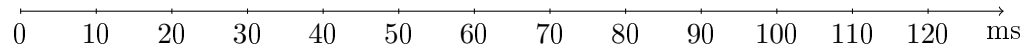
- J1 : $\langle 0,40 \rangle$
- J2 : $\langle 0,30 \rangle$
- J3 : $\langle 10,10 \rangle$
- J4 : $\langle 20,30 \rangle$

J1:

J2:

J3:

J4:



Namn: _____ Persnr: _____

3.2 stride scheduling [2 poäng*]

Man skulle kunna implementera en *stride scheduler* genom att ha alla processerna i en lista ordnad efter ett s.k. *pass value*. Den process som är först i listan väljes som den process som skall exekveras. När processen har exekverats så skall den läggas in i listan igen, på vilken position skall den läggas in?

Namn: _____ Persnr: _____

4 Virtuellt minne

4.1 gungor och karuseller [2 poäng]

Antag att vi har ett sidindelad virtuellt minne med sidstorlek 4Ki byte. Antag att varje process har fyra segment (t.ex kod, data, stack, extra) och att dessa kan vara av godtycklig men given storlekar. Hur mycket kommer kommer operativsystemet att förlora i intern fragmentering?

Namn: _____ Persnr: _____

4.2 sidindelad minne med sidor på 64 byte [2 poäng*]

Du har som uppgift att föreslå en arkitektur för en processor som skall arbeta med ett sidindelad minne där sidorna är så små som 64 byte. Processorn är en 16-bitars processor och den virtuella adressrymden skall vara 2^{16} byte.

Föreslå ett schema som använder sig av en hierarkisk sidtabell baserad på sidor om 64 byte och förklara hur adressöversättning går till.

Namn: _____ Persnr: _____

5 Filsystem och lagring

5.1 vad kan hända [2 poäng]

Antag att vi har ett enkelt filsystem utan journal där vi skriver direkt till inoder, bitmappar och datablock. Antag att vi skall skriva till en fil och behöver ytterligare ett datablock. När vi utför operationen på disk lyckas vi dock bara ändra i inoden men inte i bitmapparna eller i det datablock som vi har valt innan vi kraschar.

Om vi inte upptäcker felet när vi återstartar, vilka problem kommer vi att ha - vad kan hända?

Namn: _____ Persnr: _____

5.2 loggbaserade fs [2 poäng*]

I ett loggbaserat filsystem skriver vi alla förändringar i en kontinuerlig logg utan att göra förändringar i de redan existerande block som en fil har. Vi kommer förr eller senare att få ont om nya block och måste på något sätt återanvända block som inte längre används.

Hur håller vi reda på vilka block som kan återanvändas och hur går vi till väga?