Operating Systems ID2206
English version
2017-04-10 14:00-18:00

Name: _____

Instruction

- You are, besides writing material, only allowed to bring one self hand written A4 of notes. Mobiles etc, should be left to the guards.
- All answers should be written in these pages, use the space allocated after each question to write down your answer.
- Answers should be written in Swedish or English.
- You should hand in the whole exam.
- No additional pages should be handed in.

Grades for 6 credits

The exam is divided into a number of questions where some are a bit harder than others. The harder questions are marked with a star *points**, and will give you points for the higher grades. The exam is thus divided into basic points and points for higher grades. First of all make sure that you pass the basic points before engaging with the higher points.

Note that, of the 40 basic points only at most 36 are counted, the points for higher grades will not make up for lack of basic points. The limits for the grades are as follows:

- Fx: 21 basic points
- E: 23 basic points
- D: 28 basic points
- C: 32 basic points
- B: 36 basic points and 12 higher points
- A: 36 basic points and 18 higher points

The limits could be adjusted to lower values but not raised.

Gained points

Don't write anything here.

Uppgift	1	2	3	4	5	6	7	8	9
Max G/H	4/0	2/2	4/2	4/2	4/2	4/2	4/2	2/2	12/8
G/H									

Total number of points:

Name:_____ Persnr:_____

1 Operating systems

1.1 what happens? [2 points]

If we give the following commands after each other in a *shell*; what will the result be?

```
> mkdir foo
> cd foo
> echo "hello hello" > tomat.txt
> mkdir ../bar
> ln tomat.txt ../bar/gurka.txt
> rm tomat.txt
> cd ../
> wc -w bar/gurka.txt
```

Answer:

1.2 commands in a shell [2 points]

Give a short description of the commands below.

- `cat`
- `less`
- `ln`
- `mv`

Answer: Have a look using `man`.

Name:_____ Persnr:_____

2 Processes

2.1 what is where? [2 points]

In the code below we have allocated two arrays; which arrays and in which segments do we find them: global, stack or heap?

```
#include <stdio.h>
#include <stdlib.h>

int x = 43;

int h[] = {1,2,3,4};

int *foo(int *a, int s) {

  int *r = malloc(s * sizeof(int));

  for(int i = 0; i < s; i++) {
     r[i] = a[i] + x;
  }
  return r;
}

int main() {

  int *c = foo(h, 4);

  printf("%d \n", c[3]);

  return 0;
}
```

Answer: The two arrays are `h` that is allocated globally and `r` that is allocated on the heap in `foo()`.

Name:_____ Persnr:_____

2.2 privileged instructions [2 points*]

In a x86-architecture some intructions are priviliged and can only be executed in *Ring-0*. As an example we can take: HLT (halt), LIDT (load interupt descriptor table), MOV CR (write to Control Register). How do we use this feature to make an efficient implementation of an operating system?

Answer:

3 Scheduling

3.1 shortest time-to-completion first [2 points]

The scheduler "shortest time-to-completion first" is a optimal scheduler; what is it that it optimizes and why is it in practice not usable?

Answer:

3.2 reaction time [2 points]

When we wan to reduce the reaction time we want to preempt a job even though the job is not completed. If we choose to do this we have one parameter to set, by changing this we can improve the reaction time. Which parameter is it? How should it be set and what unwanted consequence might it have?

Answer:

3.3 real-time scheduler [2 points*]

In a real-time scheduler, jobs are described by three values: e, d och p. What is the meaning of these parameters (you don't have to remember which is which but should give the properties that describe a job).

Answer:

- e:
- d:
- p:

Name:_____ Persnr:_____

4 Virtual memory

4.1 paging [2 points]

Assume we have a virtual address that consist of a 22-bit page number and a 12-bit offset. We have a physical address space of 32 bits, encode a frame number in 20 bits and elements in a page table are 4 bytes. How large is a page and how large would a complete page table be?

Answer:

4.2 reverse page table [2 points]

If we have a physical memory that is much smaller than the virtual memory that the operating system provides it could be an idea to implement so called inverted page table. How does an inverted page table work and what benefits does it give?

Answer:

4.3 bounds [2 points*]

When we implement a segmented memory we needa bounds value that describes the size of the segment. If we do not have this we risk accessing memory outside of the segment. When we implement a paged memory there is no need for a bounds value. Why don't we need one? What prevents us from addressing outside of the page?

If there is something that prevents us from doing the wrong thing, is there someting that costs. What does it cost?

Answer:

5 Memory management

5.1 camping vacation [2 points]

When I'm out camping in Sweden different camp sites have different rules. Some tell me to place my tent anywhere but make sure that I leave a four meter distance to other tents. Other sites have divided their park into equal size lots and tell me to place my tent in lot nuymber 17.

Name:_____ Persnr:_____

What is the problem with each of the strategies and what has this to do with operating systems.

Answer:

5.2 static reallocation [2 points]

Instead of implementing a dynamic memory management we can make a static reallocation of programs when they are inserted into memory. What does it mean to do a static reallocation, what has to be done when a program is placed in memory? Give an example.

Answer:

5.3 code, data, stack ... [2 points*]

When implementing a segmented memory, one solution is for example to let the uppermost bits in a virtual address determine which segment to use. Another solution is to have separate segments for code, data and stack. If we would have only these three segments, how would the processor determine which segment to use when we do operations on the memory?

Answer:

Name:_____ Persnr:_____

6 Concurrent programming

6.1 count [2 points]

What will be printed if we execute the procedure `hello()` below concurrently in two threads? Motivate your answer.

```
int loop = 10;

void *hello() {
  int count = 0;

  for(int i = 0; i < loop; i++) {
      count++;
  }
  printf("the count is %d\n", count);
}
```

Answer:

6.2 but why [2 points]

If we have a multicore CPU it is of course an advantage to work with several threads since we can better make use of the computational power. If we only have one core it is rather pointless to divide a program up into threads, or? Kan a program divided into threads execute faster even if we only have one core to run on? Motivate.

Answer:

6.3 green treads [2 points*]

Some operating systems provide so called *green threads* that is, threads that are implemented by library procedures and that are managed by the user process. Describe an advantage and a disadvantage of green threads.

Answer:

7 File systems and storage

7.1 a regular HDD [2 points]

If a hard disk drive has a average seek time of 10 ms, a rotation speed of

Name:_____ Persnr:_____

7200 rpm (rounds per minute) and a read performance of 200 MiB/s. Then what is the average time to read a random sector on 4KiB

Answer:

Name:_____ Persnr:_____

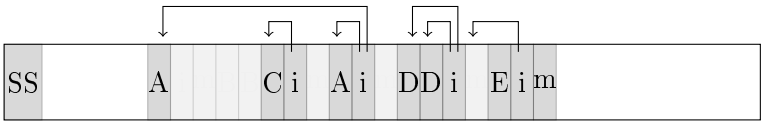
7.2 sectors to file system [2 points]

A hard disk drive consist of a a number of sectors. Describe a simple implementation of a file system and the data structures of the system can be stored on the drive.

Answer:

7.3 log based file system [2 points*]

Below you see a schematic image of a log based file system. If the system now has a shortage of space it will try to create some. How can more space be created and what will the system look like when this has been done?



Answer:

Name:_____ Persnr:_____

8 Virtualization

8.1 why [2 points]

Describe one important reason why you would like to run several virtual operating systems instead of one single.

Answer:

8.2 kernel/user mode [2 points*]

Assume that we have hypervisor running in *kernel mode* and a virtualized system running in *user mode* to protect the hypervisor. Why can we not allow the virtualized operating system to remain in *user space* when one of its processes is executing?

Answer:

Name:_____ Persnr:_____

9 Implementation

9.1 the stack [2 points]

Below we see a program that will print the content of the stack.

```
void zot(unsigned long *stop, int a1, int a2, int a3, int a4, int a5, int a6) {
  unsigned long r = 0x456;
  unsigned long *i;
  for(i = &r;  i <= stop; i++){
    printf("%p      0x%lx\n", i, *i);
  }
}

int main() {
  unsigned long p = 0x123;

  zot(&p,1,2,3,4,5,6);
 back:
  printf("  back: %p \n", &&back);
  return 0;
}
```

When executed we see the following print out. Describe the values indicated with arrows (<--).

```
0x7ffeb3331f58      0x456
0x7ffeb3331f60      0x7ffeb3331f60  <-- ??
0x7ffeb3331f68      0x3a7dbfad7df4b100
0x7ffeb3331f70      0x7ffeb3331fa0
0x7ffeb3331f78      0x400663    <-- ??
0x7ffeb3331f80      0x6   <-- ??
0x7ffeb3331f88      0x4004a0
0x7ffeb3331f90      0x123
  back: 0x400667
```

Answer:

Name:_____ Persnr:_____

9.2 sbrk() [2 points]

The procedure `malloc()` is a library routine that uses the system call `sbrk()` to allocate memory. What is the advantage for a process to use `malloc()` instead of calling `sbrk()` directly?

Answer:

9.3 pipes [2 points]

If we have two processes, one producer and one consumer, that are communicating through a so called *pipe*. How can we then prevent that the producer sends more information than the consumer is ready to receive and thereby crash the system.

Answer:

9.4 tmpfs [2 points]

What does the command below do and why would we like to do that? What is the disadvantage?

```
> sudo mount -t tmpfs tmpfs ./tmp
```

Answer:

9.5 swap context [2 points]

The program below is playing its own *context*. What does the structure `ucontext_t` contain and what will the result be when we execute the program?

```
#include <stdlib.h>
#include <stdio.h>
#include <ucontext.h>

int main() {

  int done = 0;
  ucontext_t one;
  ucontext_t two;

  getcontext(&one);

  printf("hello %d\n", done);
```

Name:_____ Persnr:_____

```
  if(!done) {
    done = 1;
    swapcontext(&two, &one);
  }
  return 0;
}
```

Answer:

9.6 read performace [2 points]

How large is the difference in read performance if we compare reading from main memory to reading from a file from a rotating hard drive (first read of file)?

- $10ns$ vs $1\mu s$
- $10ns$ vs $10ms$
- $100ns$ vs $10\mu s$
- $1\mu s$ vs $1ms$

Answer:

9.7 a header and a footer [2 points*]

When implementing a memory allocator it is common to hide a *header* just before the memory area that is allocated. In this head you can for example record the size of the are to make it easier to handle the area when *freed*. One can also use a hidden footer after the area where one can record that the area is used or free and maybe a pointer to its head. What is the advantage of having this information after the area, is it not enough with the header?

Answer:

9.8 lite bättre [2 points*]

A so called *pipe* is a simple way to send data from one process to another. It does have its limitations and a better way is to use so called *sockets*. If we instead of a pipe use a stream socket between two processes we will have several advantages. Describe two advantages that a steam socket gives us that we will not have if we use a pipe.

Answer:

Name:_____ Persnr:_____

9.9 character device [2 points*]

We can create so called *character device* and interact with it using `ioctl`. In the code below, describe what `fd`, `JOSHUA_GET_QUOTE` and `buffer` is and how the *device* could work.

```
if (ioctl(fd, JOSHUA_GET_QUOTE, &buffer) == -1) {
  perror("Hmm, not so good");
} else {
  printf("Quote - %s\n", buffer);
}
```

Answer:

9.10 delat minne [2 points*]

Processes in a Unix-system can communicate with each other over several different channels. They can use shared memory areas much in the same way as two threads in a process can share heap and global data. How can we make two processes share memory?

Answer:

Name:_____ Persnr:_____

Answer: Vi kan skapa digitala signaturer och autentisera någon.

Answer: I så kallade hårda system så har vi tidsgränser som vi garanterat måste hålla. I ett mjuk system har vi gränser som är mer eller mindre viktiga, vi kan bryta dem men vi måste notifiera användaren på något sätt så att man kan hantera situationen.

Answer: Vi måste till exempel ta hänsyn till vilken kärna en process körde på senast och i möjligaste mån låta processen köra på samma kärna får att förbättra cache-prestanda. Schemaläggaren kan ha en kö per kärna och endast flytta processer mellan kärnor om balansen blir alltför snedfördelad.

Answer: Vi kan till exempel låta processer kommunicer genom delat minne ellr *pipes*. Dessa former av kommunikation är normalt begränsade till en maskin men det distribuerade systemet skall ge oss den möjligheten oavsett var någonstans en process råkar köra.