Operating Systems ID2206

English version (only for ID2206 HT16)

2017-08-21 8:00-12:00

Instruction

- You are, besides writing material, only allowed to bring one self hand written A4 of notes.
- 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.

Name:_____ Persnr:_____

1 Operating systems

1.1 what happens? [2 points]

What is `foo` and `bar`?

```
$ ls -l
totalt 8
drwxrwxr-x 2    kalle admin  4096 dec  1 11:53 bar
-rw-rw-r-- 1    jonny angels   58 dec  1 11:53 foo
```

Answer: `foo` is a fiel and `bar` is a directory.

1.2 commands in a shell [2 points]

Give a short description of the commands below.

- `ln`
- `chown`
- `less`
- `pwd`

Answer: Have a look using `man`.

Name:_____ Persnr:_____

2 Processes

2.1 what is the problem? [2 points]

The code below might compile but we do a severe error. Which is the error and what could happen?

```
#include <stdlib.h>

#define SOME 100  // should be > 2

int *some_fibs() {

  int buffer[SOME];

  buffer[0] = 0;
  buffer[1] = 1;

  for(int i = 2; i < SOME; i++) {
     buffer[i] = buffer[i-1] + buffer[i-2];
  }
  // buffer contains SOME Fibonacci numbers
  return buffer;
}
```

Answer: The array `buffer` is allocated on the stack and will most likely be overwritten in the next procedure call.

2.2 Intel 80286 [2 points*]

In the processor 80286, that was launched in 1982, Intel had added a privileged instruction `LIDT` (Load Interrupt Descriptor Table). What does it mean that the instruction is privileged and why does this instruction need to be privileged?/

Answer:

3 Scheduling

3.1 interactive processes [2 points]

Assume that we want to implement a scheduler that gave good response time to interactive processes but we did not know which processes were interactive.

Name:_____ Persnr:_____

What would a good heuristic be to determine which processes are interactive and how could we give them better response time?

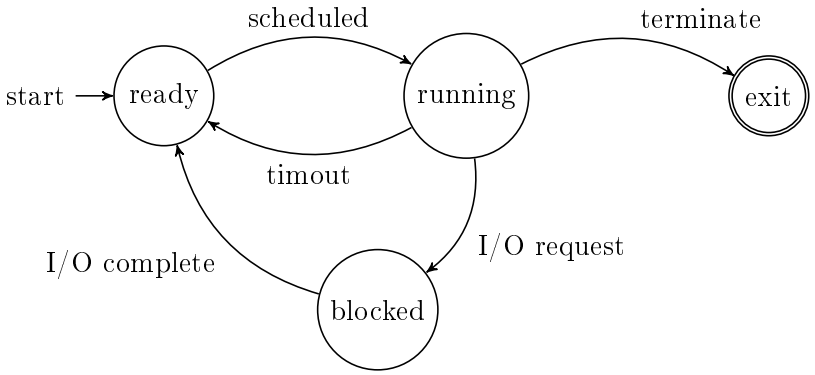
Answer: TBD

Name:_____ Persnr:_____

3.2 state diagram [2 points]

Here follows a state diagram for scheduling of processes. Enter the marked states and transitions to describe what states means and when a process is transferred between different states.

Answer:



3.3 stride scheduling [2 points*]

One could implement a *stride scheduler* by keeping all processes in list sorted by *pass value*. The process that is first in the list is the one selected for execution. When the process has executed it is inserted in the list again, at what position should it be added.

Answer:

Name:_____ Persnr:_____

4 Virtual memory

4.1 segmenting [2 points]

When we use segmentation to handle physical memory we could have problems with external fragmentation. This is avoided if we instead use paging. How is it that we can avoid external fragmentation using paging?

Answer: Since all frames are of equal size and a process can be allocated any page, a page can always be reused. No pages are too small to be used.

4.2 a tree [2 points]

When representing a *page table* a tree structure is used. Why use a tree structure, it would be faster to access an entry if the table was represented as an array with direct access to the entries. A tree will only give us one or more indirection so why use a tree?

Answer: We don't need to represent the whole table but only a fraction of the virtual address space that is used. This will give us a much smaller data structure to manage - important for a 32-bit address space and completely decisive for a 64-bit space.

Name:_____ Persnr:_____

4.3 x86_64 addressing [2 points*]

In a x86-processor in 64–bit mode a PTE contains a 40-bit frame address. This is combined with a 12 bit offset to a physical address. This is 52 bits but a process only has a 48-bit virtual memory. What advantage is there to have a 52-bit physical address.

Answer:

5 Memory management

5.1 malloc() [2 points]

In Linux (and all Unix dialects) malloc is a library procedure and not a system call. Why is it a library procedure? Would it not be quicker if we called the operating system directly, in the end it will manage the memory any way.

Answer:

5.2 who's your buddy [2 points]

Assume that we use buddy allocation and have smallest block of size 16 bytes. If we free a block of size 32 bytes that has number 0b001010, which block is then our buddy?

Answer: The buddy is a 32 byte blocka at 0x001000.

5.3 address sorted [2 points*]

Assume that we implement a memory manager and keep the free blocks in a single linked list. If we keep the list sorted on address we might find blocks that should be coalesced. What would a implementation look like and how do we identify the blocks.

Answer:

Name:_____ Persnr:_____

6 Concurrent programming

6.1 count [2 points]

If we execute the procedure `hello()` below in two threads concurrently, the result will be - yes, what will the result be? How is this possible?

```
int loop = 10;

void *hello() {
  int count = 0;

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

Answer: The variable `count` is local on the stack in each thread. Both threads will increment their variable to 10 and write the result.

6.2 thing in the heap [2 points]

If we have a multithreaded program the threads can of course read and write global variables and thus work with shared data structures. How is it with data structures that one thread allocates on the heap, can these structures be read and written to by other threads?

Answer:

Name:_____ Persnr:_____

6.3 thread local storage [2 points*]

Assume that we implement a memory manager (alloc/free) where the free list is handled using the construct below. Which advantages and possible disadvantages would this give us?

```
__thread  chunk *free = NULL;

void free(void *memory) {
  if(memory != NULL) {
    struct chunk *cnk = (struct chunk*)((struct chunk*)memory - 1);
    cnk->next = free;
    free = cnk;
  }
  return;
}
```

Answer:

Name:_____ Persnr:_____

7 File systems and storage

7.1 rpm [2 points]

There are hard drives with different performance, one thing that differ is the rotation speed. Why is the rotation speed important, what is improved and what could possibly decrease?

Answer:

7.2 what goes where [2 points]

Assume that we have simple file system without a journal where we write directly to bitmaps, inodes and data data blocks. Assume that we shall write to a file and that am additional data block is needed. Which structures are updated and which changes are made?

Answer:

Name:_____ Persnr:_____

7.3 log-based fs [2 points*]

In a log based file system we write all changes to a continuous log without doing any changes to existing blocks of a file. What is the advantage of writing new modified copies of blocks rather than do the small changes we want to do in the original blocks? If it is better, are there any disadvantages?

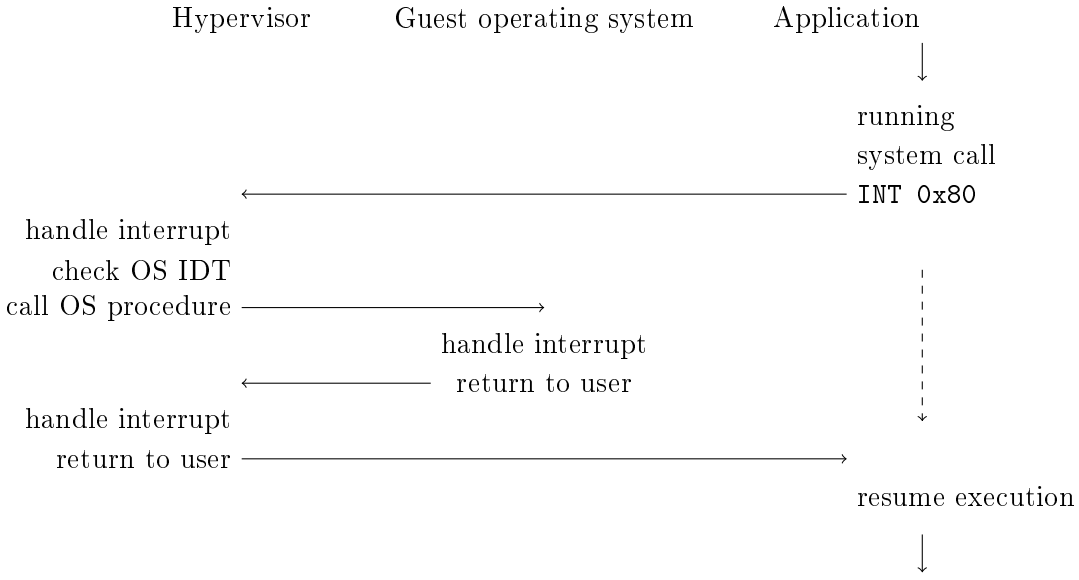
Answer:

Name:_____ Persnr:_____

8 Virtualization

8.1 a system call [2 points]

When we do a system call from a user process in a virtualized operating system, control is passed between: user process, the virtualized operating system and the hypervisor. Show in a sequence diagram what happens from the execution of `INT 0x80` to the point where control is resumed.



Answer:

8.2 set the IDT [2 points*]

When a hypervisor starts a virtualized operating system, the virtualized system will want to set the register that controls the location of the IDT. What is the problem and how is it solved?

Answer:

Name:_____ Persnr:_____

9 Implementation

9.1 memory map [2 points]

Below is a, somewhat shortened, printout of a memory mapping of a running process. Briefly describe the role of each segment marked with `???`.

```
> cat /proc/13896/maps

00400000-00401000 r-xp 00000000 08:01 1723260            .../gurka ???
00600000-00601000 r--p 00000000 08:01 1723260            .../gurka ???
00601000-00602000 rw-p 00001000 08:01 1723260            .../gurka ???
022fa000-0231b000 rw-p 00000000 00:00 0                  [???]
7f6683423000-7f66835e2000 r-xp 00000000 08:01 3149003    .../libc-2.23.so ???
    :
7ffd60600000-7ffd60621000 rw-p 00000000 00:00 0          [???]
7ffd60648000-7ffd6064a000 r--p 00000000 00:00 0          [vvar]
7ffd6064a000-7ffd6064c000 r-xp 00000000 00:00 0          [vdso]
ffffffffff600000-ffffffffff601000 r-xp 00000000 00:00 0  [vsyscall]
```

Answer: The first three segments are: code, read-only data and global data for the running process `gurka`. Then there is a segment for the *heap*. The segment marked with `lib-2.23.so` is a shard library. In the uppermost region we find the segment of the *stack*.

Name:_____ Persnr:_____

9.2 fork [2 points]

If we run the program below, what will be printed on the terminal? Why?

```
int x = 0;

int main() {

  int pid;

  pid = fork();

  if(pid == 0) {
    printf("child: x is at %p\n", &x);
  } else {
    printf("mother: x is at %p\n", &x);
    wait(NULL);
  }

  return 0;
}
```

Answer:

Name:_____ Persnr:_____

9.3 pipes [2 points]

We can easily do a `fork()` and then set up `stdin` and `stdout` for the two processes to communicate through a so called `pipe`. How can we make tow processes do the same if we do not create the process using a `fork()`. How can one process create a pipe that another process can read from?

Answer:

Name:_____ Persnr:_____

9.4 a directory [2 points]

A directory is in Linux represented in the same way as a file i.e. an inode that is pointing to a data block and the data block holds the name and identifiers of the files in the directory. This mean that we use the same system calls when we read a directory as we use when we read a file - true or false? Motivate.

Answer:

9.5 sbrk() and then what [2 points]

You can use the system call `sbrk()` to allocate more memory for the heap but how can a process return memory?

Answer:

Name:_____ Persnr:_____

9.6 a cheap operation [2 points]

Below is a extract from a program that implements the *clock algorithm*. The code shows why the clock-algorithm is cheaper compared to LRU. What should we have done in the corresponding case if we implemented LRU?

```
    :
if(entry->present == 1) {
  entry->referenced = 1
} else {
  :
}
```

Answer: The corresponding code would unlinks an entry and places it last in a double linked list. This would require several memory references, something that now can be avoided.

Name:_____ Persnr:_____

9.7 execlp() [2 points*]

In the program below we call the library procedure `execlp()` that replaced the current executing process' code. Here we use it to make call to `/bin/ls` and output both the current directory and the home directory of a user. Why will this not work?

```
int main() {

  int pid = fork();

  if(pid == 0) {

    char cwd[1024];
    getcwd(cwd, sizeof(cwd));
    printf(" This is the current directory: \n");
    execlp("/bin/ls", "/bin/ls", &cwd, NULL);

    printf(" This is your home directory: \n");
    execlp("/bin/ls", "/bin/ls", getenv("HOME"), NULL);

  } else {
    wait(NULL);
  }
  return 0;
}
```

Answer: The procedure `exexlp()` will replace the process memory segment with the code and data from the program `/bin/ls`. The code after the first call will therefore never be executed. Only the current directory will be listed.

Name:_____ Persnr:_____

9.8 sockets [2 points*]

When we want to communicate between processes we can use so called *sockets*. They come in different versions, among other `SOCK_STREAM` and `SOCK_DGRAM`. Describe the difference between these versions.

Answer:

9.9 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:_____

9.10 context [2 points*]

By the help of the library procedure `getcontext()`, a process can safe its own so called `context`. We could build a library that allowed us to create new executing threads and manually switch between these by calling a scheduler.

Why would we want to build such a library, are there any advantages? What would the disadvantages be?

Answer:

Name:_____ Persnr:_____

Answer:

Name:_____ Persnr:_____

Answer:

Answer: LANGProblemet är att anrop til tjänsten kan dubbleras och utför a mer än en gång. Man löser det genom att enbart ha s.k. idepotenta operationer dvs operationer som kan processas flera gånger utan att tillståndet förändras mer än vid första anropet.