Operating Systems ID1206

Exam

2018-01-12 14:00-18:00

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

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 24 basic points only at most 22 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: 12 basic points
- E: 13 basic points
- D: 16 basic points
- C: 20 basic points
- B: 22 basic points and 6 higher points
- A: 22 basic points and 10 higher points

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

Name:_____ Persnr:_____

1 Processer

1.1 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 42  // should be  2..47

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;
}
```

Name:_____ Persnr:_____

1.2 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]
```

Name:_____ Persnr:_____

1.3 Arghhh! [2 points]

Assume that we have a program `boba` that writes "Don't get in my way" to `stdout`. What will the result be if we run the program below and why is this the result? (the procedure `dprintf()` takes a file descriptor as argument)

```
int main() {

  int fd = open("quotes.txt", O_RDWR | O_CREAT, S_IRUSR | S_IWUSR);

  int pid = fork();

  if(pid == 0) {
    dup2(fd, 1);
    close(fd);
    execl("boba", "boba", NULL);
  } else {
    dprintf(fd, "Arghhh!");
    close(fd);
  }
  return 0;
}
```

Name:_____ Persnr:_____

1.4 list of free blocks [2 points]

If we when implementing `malloc()` and `free()` choose to save the free blocks in a linked list that is ordered by their address, we will have a certain advantage. When we free a block we can insert it in the list and perform an operation that reduces the external fragmentation. What can we do and why is it an advantage to have the blocks order by address? Show with a drawing what information is used and how the operation is performed.

Name:_____ Persnr:_____

1.5 intern paging [2 points*]

When we implement memory internally for a process (for example in malloc()) we us a form of segmentation. This is why we could have problem with external fragmentation. If it's better to use paging why do we not use it when we implement internal memory management?

Name:_____ Persnr:_____

1.6 context [2 points*]

By the help of the library procedure `getcontext()`, a process can save 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?

Name:_____ Persnr:_____

2 Communication

2.1 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);
}
```

Name:_____ Persnr:_____

2.2 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.

Name:_____ Persnr:_____

2.3 name space [2 points*]

Below is code were we open a socket and use the name space `AF_INET`. We will then be able to address a server using a port number and IP-address. There are other name spaces that we can use when working with sockets. Name one and decribe its advantages and disadvantages it might have.

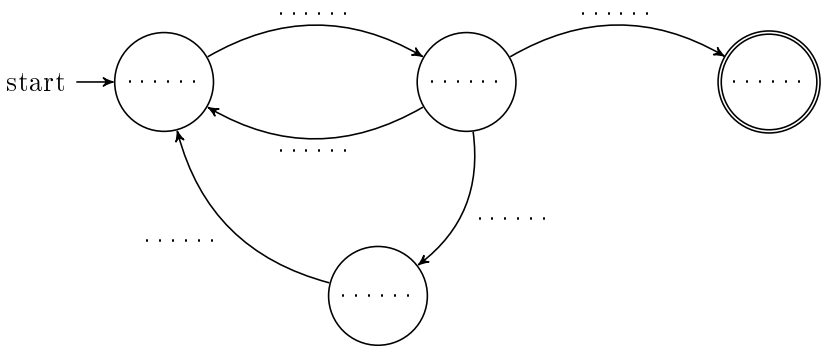
```
struct sockaddr_in server;
server.sin_family = AF_INET;
server.sin_port = htons(SERVER_PORT);
server.sin_addr.s_addr = inet_addr(SERVER_IP);
```

Name:_____ Persnr:_____

3 Scheduling

3.1 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.



3.2 reaction time [2 points]

When we want 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?

Name:_____ Persnr:_____

3.3 rate monotonic scheduling [2 points*]

A real time scheduler based on "Rate Monotonic Scheduling" (fixed priority where priority is determined by periodicity). is a relative simple scheduler. If we assume that deadlines are equal to the full peiodicity, how can we then describe the load of a system?

Do we have any guarantees that the scheluling will work i.e. that no deadlines are missed?

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? Is there something that we risk?

Name:_____ Persnr:_____

4.2 almost right [2 points]

Below is a extract from a program that implements *Least Recently Used* (LRU). The code shows why LRU is expensive to implement and why one probably instead choose to approximate this strategy. How could we approximate the algorithm and which consequences would this bring? Could part of the algorithm be implemented in hardware?

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

    if(entry->next != NULL) {

      if(first == entry) {
        first = entry->next;
      } else {
        entry->prev->next = entry->next;
      }
      entry->next->prev = entry->prev;

      entry->prev = last;
      entry->next = NULL;

      last->next = entry;
      last = entry;
    }
} else {

  :
}
```

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.

Name:_____ Persnr:_____

5 File systems and storage

5.1 list the content of a directory [2 points]

If we want to list the content of a directory we can use the library procedure `opendir()`. Which information can we access directly form the structure pointed to by `entry` in the code below? Describe three important properties. Which information can we not find and where could this information be found?

```
int main(int argc, char *argv[]) {

  char *path = argv[1];

  DIR *dirp = opendir(path);

  struct dirent *entry;

  while((entry = readdir(dirp)) != NULL) {

    //  what information do we have?

  }
}
```

5.2 remove a file [2 points]

If we us the command `rm` we will not remove a file, rather remove a hard link to a file. When is the file it self removed? How is this handled?

Name:_____ Persnr:_____

5.3 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?