Operating Systems ID1206

(ID2200/06 6hp)

Exam: example of new structure

HT19

Instruction

- You are, besides writing material, only allowed to bring one self hand written A4 of notes. The notes are handed in and can not be reused.
- 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 and the hand written page of notes. No additional pages should be handed in.

Grades

- Fx: 7 points of 10 in the basic part
- E: 8 points of 10 in the basic part

Higher grade is given based on the result on the second part

- D: two passed
- C: three passed
- B: four passed
- A: all passed

Name:_____ Persnr:_____

1 stack or heap [2 points]

What is done in the procedure below and where should `gurka` be allocated? Why? Complete the code so that `gurka` is allocated space.

```
int *tomat(int *a, int *b) {
  // allocate room for gurka


  *gurka = *a + *b;
  return gurka;
}
```

Name:_____ Persnr:_____

2 fork() [2 points]

What is printed when we run the program below, what alternatives exist and why do we get this result?

```
int global = 17;

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

Name:_____ Persnr:_____

3 `__sync_val_compare_and_swap()` [2 points]

We can implement a spin lock in GCC as shown below. The lock is implemented using a machine instruction that atomically will read the content of a memory location and, if it is equal to our requirement, replace it with a new value. In the implementation below we represent an open lock with the value 0; if the lock is open we write a 1 in the location and return 0, otherwise we return the value found (that then should be 1).

Assume that we use the lock to synchronize two threads on a machine with only one core; what is then the disadvantage that we will have? How could we mitigate the problem?

```
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;
}
```

Name:_____ Persnr:_____

4 a stack, a bottle and.. [2 points]

You have written the program below to examine what is on the stack.

```
void zot(unsigned long *stop ) {
  unsigned long r = 0x3;
  unsigned long *i;
  for(i = &r;  i <= stop; i++){ printf("%p          0x%lx\n", i, *i);  }
}

void foo(unsigned long *stop ) {
  unsigned long q = 0x2;
  zot(stop);
}

int main() {
  unsigned long p = 0x1;
  foo(&p);
 back:
  printf("  p: %p \n", &p);
  printf("  back: %p \n", &&back);
  return 0;
}
```

This is the print out. Explain what is found at the locations indicated by arrows.

```
0x7ffca03d1748      0x3
0x7ffca03d1750      0x7ffca03d1750
0x7ffca03d1758      0xb93d7906926a7d00
0x7ffca03d1760      0x7ffca03d1790        <-------
0x7ffca03d1768      0x55cdac31d78c        <-------
0x7ffca03d1770      0x7ffca03d17d8
0x7ffca03d1778      0x7ffca03d17b0
0x7ffca03d1780      0x1
0x7ffca03d1788      0x2
0x7ffca03d1790      0x7ffca03d17c0
0x7ffca03d1798      0x55cdac31d7c2
0x7ffca03d17a0      0x55cdac31d810
0x7ffca03d17a8      0x12acac31d5f0
0x7ffca03d17b0      0x1
  p: 0x7ffca03d17b0
  back: 0x55cdac31d7c2
```

Name:_____ Persnr:_____

5 library call vs system call [2 points]

An operating system that implements POSIX should provide specified functionality to a user process. Is this provided by system calls, library procedures or a combination of both? Explain the difference between system calls and procedure calls and which parts belong to the operating system.

Name:_____ Persnr:_____

6 from one to the ...[P/F]

Assume that we have two programs, `ones` and `add2`, implemented as bellow. The call to `scanf("\%d", \&in)` will read from `stdin` and parse a number that is then stored in `\&in`. The procedure either returns 1, if it manages to read number, or `EOF`. The call to `printf()` will write the number to `stdout`.

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

int main() {

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

  return 0;
}
```

```c
/* 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;
}
```

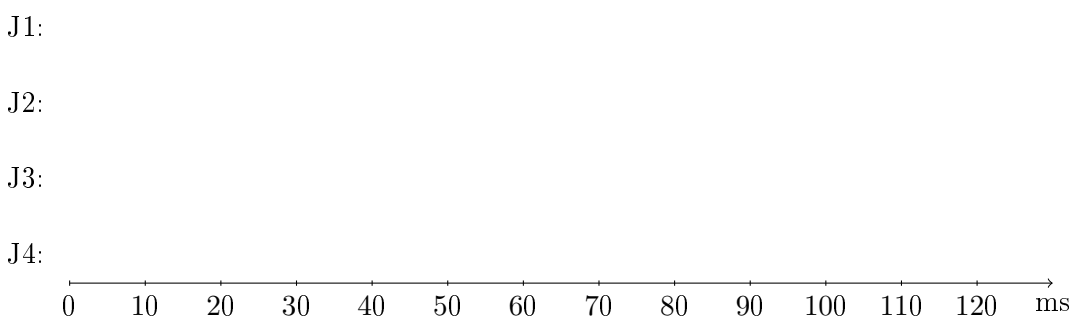
You have a Linux computer and all possible programs. How would you in the simplest possible way make the output from the first program, `ones`, be read by the the other program `add2`.

Name:_____ Persnr:_____

7 Scheduling [P/F]

Assume that we have a scheduler that implements *shortest job first*. We have four jobs described below as ⟨*arrive at*, *execution time* ⟩ in ms. Draw a time diagram and specify the turnaround time for each of the jobs.

- J1 : ⟨0,40⟩
- J2 : ⟨0,30⟩
- J3 : ⟨10,10⟩
- J4 : ⟨20,30⟩



Name:_____ Persnr:_____

8 you win some,you loose some [P/F]

Assume that we have a paged virtual memory with a page size of 4Ki byte. Assume that each process has four segments (for example: code, data, stack, extra) and that these can be of arbitrary but given size. How much will the operating system loose in internal fragmentation?

Name:_____ Persnr:_____

9 paged memory with 64 byte pages [P/F]

You have been asked to propose an architecture for a processor that should have a paged virtual memory with the page size as small as 64 byte. The processor is a 16 bit processor and the virtual address space should be 2^{16} bytes.

Propose a scheme that uses a hierarchical page table based on pages of 64 Ki byte and explain how the address translation is done.

Name:_____ Persnr:_____

10 log-based fs [P/F]

In a log-based file system we write all changes in a continuous log without changing the existing data blocks that has been allocated to a file. We will sooner or later run out of blocks and need to reclaim blocks that are no longer used.

How do we keep track of which blocks that can be reused and what do we do to reclaim the blocks?