Name _____ Matriculation no. _____ Tutorial no. _____

Karlsruhe Institute of Technology

Operating Systems 2013/14
Assignment 2

Prof. Dr. Frank Bellosa
Dipl.-Inform. Marius Hillenbrand
Dipl.-Inform. Marc Rittinghaus

Submission Deadline: Monday, December 2nd, 2013 – 9:30 a.m.

In this assignment you will study system calls and processes.

Please print out the pages containing **T-Questions** and answer them on your printout. Clearly mark every page with your name, matriculation number, and **tutorial number**. Simply put it in the mailbox in the basement of building 50.34 (Info-Neubau).

P-Questions are programming assignments. Download the provided tarball from the VAB and make sure to use the included templates and Makefiles. Do not fiddle with the compiler flags. Submission instructions can be found in the first assignment.

Any assignment handed in after its deadline will be ignored!

T-Question 2.1: Processes

a. What is the difference between a program and a process? **1 T-pt**

b. When a subprocess exits, it may become a zombie. What is a zombie and what needs to be done for this not to happen? **1 T-pt**

c. Process A creates process B which in turn creates process C. In a System-V based system: What is C's parent after B was killed? **1 T-pt**

d. What is the context of a process? Name at least 4 properties. **2 T-pt**

Name _____ Matriculation no. _____ Tutorial no. _____

e. One difference between a process that is running and one that is not is where the context is kept. Where is the context stored respectively? Which data structures are involved? **2 T-pt**

T-Question 2.2: Threading Models

a. Which of the following objects are shared by all threads of a multi-threaded process, which exist per thread? (correctly marked: 0.5P, not marked: 0P, incorrectly marked: -0.5P) **3 T-pt**

shared	per thread	
☐	☐	Stack
☐	☐	Heap
☐	☐	Code
☐	☐	Instruction pointer (Program counter)
☐	☐	Open files
☐	☐	Register contents

b. Enumerate two disadvantages of the Many-to-One model. **1 T-pt**

c. Why does a thread switch to another thread of the same application normally take longer in the One-to-One model than in the Many-to-One model? **1 T-pt**

Name _____ Matriculation no. _____ Tutorial no. _____

T-Question 2.3: Scheduling Basics

a. Which hardware is required for preemptive scheduling? **1 T-pt**

b. When does the scheduler get active with preemptive and with non-preemptive scheduling? Give an example for each case. **1 T-pt**

c. Briefly state the differences between the CPU scheduler and the dispatcher. **1 T-pt**

d. Four independent jobs are run in parallel on a computer with a quadcore (4-core) processor. It is possible that the jobs run 4 times as fast as on a single core, but it is also possible that they run only as fast as on a single core. Explain why, and what the characteristics of each job are in each scenario. **2 T-pt**

P-Question 2.1: System calls

a. The file descriptors 0, 1, and 2 denote the standard input, standard output, and standard error. Write two functions that print the given arguments a, b, and c, separated by spaces and terminated by a newline to standard output. Format `a` as a character, `b` as a decimal number and `c` as an octal number (mind the signed/unsigned). Use the library call `printf` in the first one (`print3`) and the syscall write in the second (`write3`)! Do not use any functions of the `sprintf`-family. **3 P-pt**

```
void print3( char a, int64_t b, uint8_t c );
void write3( char a, int64_t b, uint8_t c );
```

b. Write a function which returns the current date and time as a string (`char*`). Use `gettimeofday()`. The format should be "DD.MM.YYYY - hh:mm:ss", with D, M, Y being the day, month, and year, and h, m, s being the hour, minute, and second respectively (e.g., 01.01.2012 - 23:42:07). Hint: Keep in mind that the stackframe is freed after returning from the function. Add a comment which explains where and how the memory for the string is freed. Feel free to use any library function or syscall in your `getTime`. **2 P-pt**

```
char* getTime();
```

P-Question 2.2: Processes and Threads

a. Write a function that creates `n` subprocesses. Each child process shall print "<pid>" and then exit. The parent shall print "I am the parent process of: <pid>". To clarify: $2 * n$ messages, <pid> denotes the PID of the child. Take care of zombies! **2 P-pt**

```
void spawnChildren( unsigned int n );
```

b. Write a function that creates `n` threads. Each thread shall print "<i> <tid>" and then exit. The parent shall print "I just created thread <i>". To clarify: $2 * n$ messages, <tid> denotes the TID of the child, <i> runs from 1 to n. Read the man pages for `pthread_create`, `pthread_join`, and `gettid` before you start. **2 P-pt**

```
void spawnThreads( unsigned int n );
```

P-Question 2.3: A Simple Shell

Put the implementation of your shell in the (yet empty) directory `asst2-shell`. Feel free to create source and header files as you like, but always separate declarations from definitions (the former are to be put into a header file, the latter into a source file). Copy and adapt the Makefile we provided to ease the build process of your shell.

a. Write a simple command line shell with the following features: **5 P-pt**

- Start programs that the user can specify by typing the program's full path and name (i.e., /usr/bin/who). Do not use `system`.
- Make it possible to pass commandline arguments to the program.
- The user should return to the command prompt after exiting the called program.

b. Modify the shell to additionally print the exit status of the previously started/exited program. **2 P-pt**

c. Modify the shell to additionally print how long the called program was running in the format "hh:mm:ss" (hours, minutes, seconds). **1 P-pt**

Total: 17 T-pt 17 P-pt