Karlsruhe Institute of Technology

Operating Systems 2018/19
Solutions for Assignment 1

Prof. Dr. Michael Beigl
Mathias Gottschlag, M.Sc.
Jonas Kiemel, M.Sc.

T-Question 1.1: Basics

a. What is the most important piece of software running on a computer, the operating system or applications? Why? **1 T-pt**

Solution:
Unfortunately for the staff of the operating systems group, the user applications are the most important piece of software running on a computer. The operating system's task is to provide a good execution environment for the applications that the user runs to get his work done. It is the operating systems job to stay in the background and only become active when necessary (service request, hardware management, protection, etc.).

b. Why is abstraction a central task of an operating system? **1 T-pt**

Solution:
Through abstraction the operating system hides implementation details of the hardware from the applications and thus eases application development, increases application compatibility and improves reliability by taking away complexity.

c. As computers have become cheap, today most devices (desktops, tablets, smartphones, etc.) that are running an operating system are used by a single user only. Why is protection still as important? **1 T-pt**

Solution:
Protective measures in the operating system such as the separation into user mode and kernel mode, or the isolation of processes through dedicated address spaces increases the reliability of the system. Without proper protection a buggy application might inadvertently crash the whole system, leading to data loss.

Furthermore, computers are highly interconnected today and thus permanently work on data that originates from untrusted sources (e.g., a browser displaying a website). The operating system helps in this scenario to protect the user's data from leaking or being compromised by attackers.

T-Question 1.2: The User-Kernel Boundary

a. Are the following statements true or false? (correctly marked: 0.5P) **4 T-pt**

true	false	
☐	■	The operating system kernel always runs in the background.
☐	■	The trap instruction should be privileged.
■	☐	Turning off interrupts should be privileged.
☐	■	Interrupts are synchronous to code.
☐	■	System call parameters may be passed via the kernel stack.
■	☐	System call parameters may be passed in registers.
■	☐	A system call is a voluntary kernel entry.
☐	■	Interrupts may only happen in the context of the kernel.

b. Why must the kernel carefully check system call parameters? **1 T-pt**

Solution: *System call parameters are specified by user applications and thus cannot be trusted because the applications could be buggy or malicious.*

c. What test does the kernel perform when receiving the address of a buffer (e.g., to write the contents of a file to) as a system call parameter? **1 T-pt**

Solution: *The kernel ensures that the address lies in the user address range. That way, the user cannot fool the kernel into overwriting its own data structures or leaking critical system private information to the user process.*

d. Consider the following disassembly of a function in the `ntdll.dll` system library of a 64-bit Windows 8.1. What purpose does this function serve? What is the meaning of the number marked in bold? **1 T-pt**

```
; NtCreateProcess
.text:0000000180092120  4C 8B D1        mov     r10, rcx
.text:0000000180092123  B8 AA 00 00 00  mov     eax, 0AAh
.text:0000000180092128  0F 05           syscall
.text:000000018009212A  C3              retn
```

Solution:
The disassembly shows the Windows NtCreateProcess() system call (0.5 pt). The marked number is the system call number (0.5 pt).

The function is called by an application according to the Microsoft 64-bit standard calling convention, which passes the first four arguments in the registers RCX, RDX, R8, and R9. The first line moves the first argument, because the system call dispatcher in kernel mode expects the first argument to be passed via r10. In the second line, the system call number is written to the eax register. The function then uses the syscall instruction to jump into kernel mode to the system defined system call dispatcher. The dispatcher then jumps to the kernel's system service routine specified in eax (NtCreateProcess).

Total: 10 T-pt