VE482 — Introduction to Operating Systems

Homework 7

Manuel — UM-JI (Fall 2019)

Non-programming exercises:

- Write in a neat and legible handwriting
- Clearly explain the reasoning process
- Write in a complete style (subject, verb and object)

Progamming exercises:

- Write a README file for each program
- Upload an archive with all the programs onto Canvas

Ex. 1 — *Page replacement algorithm*

In this exercise we consider the WSClock page replacement algorithm with a τ value of two ticks. The system state is given as follows.

Page	Time stamp	Present	Referenced	Modified
0	6	1	0	1
1	9	1	1	0
2	9	1	1	1
3	7	1	0	0
4	4	0	0	0

1. Explain the content of the new table entries if a clock interrupt occurs at tick 10.
2. Due to a read request to page 4 a page fault occurs at tick 10. Describe the new table entry.

Ex. 2 — *Minix 3*

The goal of this exercise is to understand and implement system calls.

1. In which files are:
   a) the constants with number and name for the system calls?
   b) the names of the system call routines?
   c) the prototypes of the system call routines?
   d) the system calls of type "signal" coded?
2. What problems arise when trying to implement a system call `int getchpids(int n, pid_t *childpid)` which "writes" the pids of up to n children of the current process into `*childpid`?
3. Write a "sub-system call" `int getnchpid(int n, pid_t childpid)` which retrieves the n-th child process.
4. Using the previous sub-system call, implement the original `getchpids` system call. The returned `int` value corresponds to the number of pids in *childpid, or `-1` on an error.
5. Write a short program that demonstrate the previous system calls.
6. The above strategy solves the initial problem through the introduction of a sub-system call.
   a) What are the drawbacks and benefits of this solution?
   b) Can you think of any alternative approach? If yes, provide basic details, without any implementation.

Ex. 3 — *Research*

Write about a page on the topic of the `ext2` filesystem. Do not forget to reference your sources.

Ex. 4 — *Simple questions*

1. If a page is shared between two processes, is it possible that the page is read-only for one process and read-write for the other? Why or why not?
2. A computer provides each process with 65,536 bytes of address space divided into pages of 4096 bytes. A particular program has a text size of 32,768 bytes, a data size of 16,386 bytes, and a stack size of 15,870 bytes. Will this program fit in the address space? If the page size were 512 bytes, would it fit?
3. When both paging and segmentation are being used, first the segment descriptor is found and then the page descriptor. Does the TLB also need a two-levels lookup?