CSU0033 Operating Systems, Homework 3

Department of Computer Science and Information Engineering
National Taiwan Normal University

Mar. 27, 2025

As always, you are encouraged to ask and help answer questions on Moodle. We are CSU0033 community!

Contents

1 Virtual memory (60 points)

1.1 The segmentation approach (15 points)

Given a 256-byte address space and a 4KB physical memory, answer the following question. Here the address is represented in hex number.

Segment	Base	Size (byte)	Grows Positive?
$Code_{00}$	`0x111`	32	1
$Heap_{01}$	`0x200`	50	1
$Stack_{11}$	`0x300`	14	0

1. Translate the virtual address `0x71` to its physical address. For address translation, write your answer in hex number, or write ‘segmentation fault’ if you think the translation is illegal. No need to consider the data size at each address.

2. Translate the virtual address `0x11` to its physical address. For address translation, write your answer in hex number, or write ‘segmentation fault’ if you think the translation is illegal. No need to consider the data size at each address.

3. What is the size of the current external fragmentation between the code segment and the heap segment?

1.2 The paging approach (35 points)

1. True or False: Without using a TLB, a two-level paging approach cannot make correct address translation.

2. Use your own words to explain the purpose of the valid bit in a PTE, in a TLB entry, and in a PDE, respectively.

3. Suppose a 32-bit computer with the 4-KB page size. Suppose the address space of a process is 4GB and each PTE is 32-bit long. Now, assume that a process only allocated one page for its code segment, one page for its heap segment, and two pages for its stack segment. Comparing with the linear-page-table approach, what would be the maximum possible space-saving in the physical memory if we use 2-level paging instead? Assume the page table directory is also of the one-page size, and that there is no restriction on where a physical page can be placed in memory.

4. Consider the following TLB. Suppose the page size is 4KB, the address space is 16KB, and the physical memory is 128KB. Answer the following questions. Write ‘segmentation fault’ if you think the address or the translation is illegal. Write ‘TLB miss’ if you think the virtual address is legal but the translation is not currently available in TLB.

   (a) For ASID 0, Translate the virtual addresses `0x012a` to its physical address.

   (b) For ASID 1, Translate the virtual addresses `0x5123` to its physical address.

VPN	PFN	valid	prot	ASID
0x2	0x13	1	rwx	0
0x1	0x13	1	rwx	1
0x1	0x08	1	rwx	0
0x3	0x1d	1	r-x	0
0x1	0x0d	0	r-x	2
0x0	0x10	1	rw-	0
0x3	0x0f	0	rwx	1

1.3 Additional question (10 points)

Use the free version of ChatGPT to get its answer regarding the pros and cons of the segmentation approach and the paging approach. Show its answer, and then write your opinion on the quality of the answer. To be specific, explain in which ways you think the answer is good, and in which ways you think the answer can be further improved.

2 Working with xv6 (40 points + 10 bonus points)

In xv6, we can press `Ctrl` + `p` to print the PID information of the current processes in the system. This functionality may help you in working with the following tasks.

2.1 Showing when a background process finished execution (40 points)

In the original xv6, if we put a process into the background, there will be no sign that the process has finished execution. Now make some changes to the kernel, to create the following behavior: Whenever a background process finished execution, the shell will print its PID and its name (no need to print its arguments). Your implementation should not print such information for a foreground process; print it only for a background process.

Here is an example execution:

```
$ sleep 60
$ sleep 60 &
$ echo Hi > out.txt
$ 5 Done  sleep

$
```

In this example, the first run of `sleep` created a foreground process, and the shell waited for it. The second run of `sleep` created a background process, and the shell did not wait for it and it allowed us to run some other programs. Then after our `echo` program has finished, the second `sleep` finished, and the kernel printed it after the fourth `$` symbol. Note that the fourth `$` symbol appeared just because by that time `echo` has finished. The fifth `$` appeared because I pressed `Enter` after I saw that the second `sleep` finished.

Submit your modification as a code patch, and explain your design.

2.2 Optional task (bonus 10 points)

In this bonus challenge, you are asked to create a configurable notification for your background sleep process. Name your new sleep program osleep.c and implement the following behavior:

- If you run osleep with argument `-s` (slient) and `&`, the program will enter sleep in the background and will sliently finish upon wakeup.
- If you run osleep with argument `-v` (verbose) and `&`, the program will enter sleep in the background and will report its PID and name, just like what you did for the task in Section 2.1.

Here is an example execution:

```
$ osleep -s 60 &
$ osleep -v 60 &
$ echo Hi > out.txt
$ 5 Done  osleep

$
```

Submit your code patch and explain your design. A way to implement this is to re-factor your code using a customized system call.