Operating Systems

Tutorial 10

Fabian Klopfer

25. Januar 2021

Intro

- Pingo Polls
- Solution sheet 8
- Preview sheet 9

Inofficial exam-relevant exercises list

Union of tasks & questions considered relevant by Max & Fabian.

Official list/sample exam to follow soon.

Comprehension questions:

- 1.1.1, 1.1.3-4, 2.1.2-4, 2.1.6, 2.4.3-5, 2.4.7, 3.1.1, 3.1.3, 4.1.1-4, 4.1.7, 5.1.1-2, 5.1.4-9, 6.1.2, 6.1.4-8, 7.1.2, 7.1.5, 7.1.7

Exercises:

- 1.4, 2.2, 2.3, 3.3, 3.4, 4.3, 4.4, 4.5, 5.2, 5.3, 5.4, 5.5, 6.2, 6.3, 6.4, 6.5, 7.2, 7.3, 8.2, 8.3, 8.4

Exercise sheet 8

Exercise 1: Comprehension Questions I

1. Does active waiting work for cooperative scheduling?
   No, as control has to be passed.
   Cooperative scheduling + active waiting $\Rightarrow$ nothing else executes $\Rightarrow$ resources won't be released $\Rightarrow$ Deadlock
2. Does the solution with active waiting and `turn` variable in Figure 2.23 in [1] with two CPUs and shared memory?
   Yes, simply use one number per process (e.g. process 5 is allowed to run when `turn = 5`
3. Can the priority inversion problem occur with threads in user space?
   Can only occur with preemptive scheduling. (No)

Exercise 1: Comprehension Questions II

4. Can race conditions occur if all threads are executed on a single core?
   Yes! When a thread/process gets interrupted and the next one uses the unfinished state
   A wants to remove an element from the list and gets interrupted before decrementing the counter (e.g. length variable).
   B wants to add something reads the wrong length and inserts at wrong position.

5. At an intersection, four cars are each waiting for the car to their right to enter the intersection. Is this a deadlock?
   Obviously. Circular wait.

6. Under what circumstances can resource traces be diagonal? (Figure 6.8 in [1])
   When enough resources are available such that there is a schedule where processes don't need to wait for resources to be released.

Exercise 1: Comprehension Questions III

7. How can the scheme of resource trajectories (Figure 6.8 in [1]) be extended to any number of processes?
   One axis/dimension per process.

8. In a system with two processes and three instances of a resource, can there be a deadlock if each process requires two instances of the resource? Why?
   No. One process gets two and executes, the other waits. When the former finishes the latter gets the resources and can execute.

Exercise 2: Deadlocks I

1. The four criteria for deadlocks are necessary for deadlocks, but not sufficient.
   Conditions:
   - No Preemption
   - Hold and wait
   - Mutual Exclusion
   - Circular Wait

   Give an example of a case that meets the four criteria but does not cause a deadlock.

Exercise 2: Deadlocks II

Non-preemptive scheduling. Resources are thread safe using MutEx.
Processes A, B, C.
Resource types S (2 instances), R (1 instance).
A requests R, gets it
B requests S, gets it
C requests S, gets it
B requests R, blocked
A requests S, blocked
B waits for A, A waits for B (or C) $\Rightarrow$ Circular wait. 1-3 by assumption.
When C executes and releases it's instance of S, A can continue then B.

2. Specify a condition under which the four criteria are necessary and sufficient for deadlocks.
   When resources are unique. There is no way to break the circular wait then.

Exercise 3: Amdahl's Law I

T: total execution time of program sequentially,

B: time to execute sequential parts,

$T - B$: time to execute parallel parts sequentially,

N: number of CPUs.

Total execution time on N CPUs: $T(N) = B + (T - B)/N$.

Total speed-up $T/T(N)$.

Speed-up factor: $S = \frac{1}{B+(1-B)/N}$

Exercise 3: Amdahl's Law II

1. A program's sequential part is 1 %. By what factor is the program accelerated when executed on 61 CPUs?
   $S = 1/\big(0.01 + (1 - 0.01)/61\big) = 38.125.$
2. To which value does the speed-up S converge for $N \to \infty$?

$$\lim_{N \to \infty} \frac{1}{B + (1 - B)/N} = \frac{1}{B}$$

3. Amdahl's rule does not take into account the cost of communication between CPUs. Extend the expression for the speed-up S to include communication costs C.

$$S = \frac{1}{B + (1 - B)/N + C}$$

So What is C? Load balancing of scheduler, locking/waiting, message passing, ...

Exercise 3: Amdahl's Law III

4. Two implementations of an algorithm have the same sequential fraction of $B = 0.001$, but differ in their communication costs of $C_1 = 0.0001N$ and $C_2 = 0.001 \ln(N)$. Determine the number of CPUs N with maximum speed-up S for both implementations.
   Extremum is a 0 of the derivative wrt. N, and minimize denominator to maximize:
   First $C_1 = 0.0001$:

$$\frac{\partial}{\partial N} B + (1 - B) \cdot N^{-1} + 0.0001N = 0$$

$$-(1 - B)N^{-2} + 0.0001 = 0$$

$$N^{-2} = \frac{0.0001}{1 - B}$$

$$N^2 = 10000 \cdot (1 - B)$$

$$N = 100 \cdot \sqrt{1 - B}$$

Exercise 3: Amdahl's Law IV

Now for $C_2 = 0.001 \ln(N)$:

$$\frac{\partial}{\partial N} B + (1 - B) \cdot N^{-1} + 0.001 \ln N = 0$$

$$-(1 - B)N^{-2} + 0.001N^{-1} = 0$$

$$1000N = \frac{N^2}{1 - B}$$

$$(1 - B)(1000N) = N^2$$

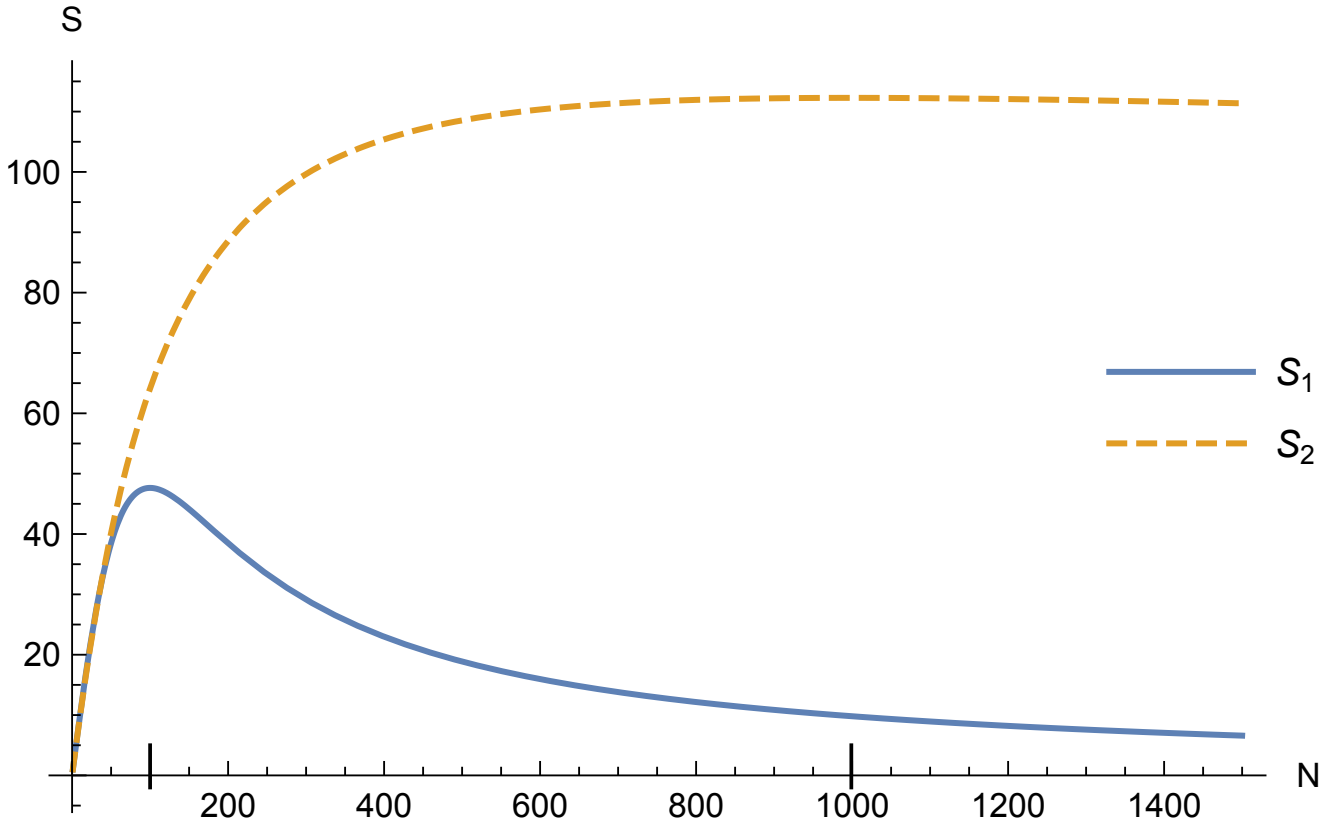
$$(1 - B) = \frac{N}{1000}$$

$$N = 1000 \cdot (1 - B)$$

100 and 1000 depending on how much is parallelizable.

Exercise 3: Amdahl's Law V

5. Plot the speed-up S as a function of the number of CPUs N for both algorithms and $1 \leq N \leq 1500$.



Exercise 3: Amdahl's Law VI

6. How do the two implementations differ qualitatively in the change in their speed-up when they use more CPUs than is optimal for them?
   The larger the communication costs, the larger the penalty when using more CPUs.

Exercise 4: Race Conditions I

What is the value of `total_count` (depending on `n`) after a call to `total`?

0

The function `total` is executed in several threads simultaneously. Describe the race condition in the program snippet. Which variables are affected? What is the effect of the race condition?

```
const int n = 5000;
int total_count;

void total(void)
{
    int count = 0;
    for (; count < n; ++count)
    if (count % 2)
    --total_count;
    else
    ++total_count;
}
```

Exercise 4: Race Conditions II

Threads may read while another thread is in the write process.
I.e. the thread changes the value in a register but hasn't stored the result back to memory
Another thread reads the lower value and overwrites the result
i.e. instead of 2 increments we only have one
Possible values: $\pm\frac{n}{2} - 1 \cdot |threads|$

```
const int n = 5000;
int total_count;

void total(void)
{
    int count = 0;
    for (; count < n; ++count)
    if (count % 2)
    --total_count;
    else
    ++total_count;
}
```

Exercise 4: Race Conditions III

Fix the race condition by using a semaphore. To do this, you can use a semaphore `int semaphore;` and the functions `void up(*int)` and `void down(*int)`.
Show code.

Exercise sheet 9

Exercise 1 (comprehension questions) $(8 \cdot 1 = 8$ points)

1. Name a device that is both an input and an output device.
2. Why can the use of a direct memory access (DMA) controller be useful even if it is slower than the CPU and memory access time?
3. Why are files first cached in a spooler folder before being output to a printer?
4. Give an example of a situation where a driver has to be reentrant.
5. To increase system reliability, drivers can also be implemented in user space. In that case, they must use system calls to read and write controller registers and buffers. Name one disadvantage of this approach.
6. How many EiB (ExbiByte) fit into one YiB (YobiByte)?
7. What is the difference between a symbolic link (symlink, soft link) and a hard link?
8. Are the `C` `printf` and `scanf` library functions implemented in user space or in kernel space?

- easy
- How fast are IO devices?
- Think about Deadlocks
- ex. many examples
- Mode switch vs. memory access
- do the math
- Think of the Inode level? What does a symlink do? What does a hard link do? (shallow vs deep copy)
- Is the C library (glibc or libc) part of the kernel?

Exercise 2 (direct memory access) $(2 + (2 + 1 + 1) = 6$ points)

1. A DMA controller has five channels that it can alternately serve. It can request a 32-bit word every 50 ns, processing the response takes the same amount of time. What is the minimum speed (in $\mathrm{GiB\,s^{-1}}$) that the bus must have in order to not delay the DMA controller?

2. In a computing system, let a (in ns) be the time to acquire the bus, and b be the time to transmit a word, where $a \geq b$. The DMA controller supports both word mode ("cycle stealing") and block mode ("burst mode"). The bus must be acquired to read or write a word.

   a) Calculate the time to transfer n words from a device controller to memory for word mode and block mode.

   b) Which mode is faster?

   c) Calculate the minimum speed-up in the limit $n \to \infty$.

   Note:
   In *word mode*, the DMA controller proceeds as follows: It sends a command to the device controller to transfer a word. The device controller transfers the word to memory and sends a word to the DMA controller for confirmation. The whole process is repeated for each word to be transferred. In *block mode*, the DMA controller sends a word to the device controller to start the transfer. The device controller occupies the bus, transfers all words to memory, and acknowledges the transfer to the DMA controller.

- word size / latency = bit/ns
- command to + read from + write to for word mode.
  command to + reserve once, n reads + ack
- trivial
- fraction of the fromulas from a with n to infinity.

Exercise 3 (program analysis, file systems) $(4 \cdot 1 = 4$ points)

In the C program below, the comments have been lost. Analyze the program and describe how it works:

1. Which POSIX system call is being used?
2. Find out what the function of this system call is. Describe it briefly.
3. What do the command line arguments do?
4. What does the program do?

```
#include <ftw.h>
#include <stdio.h>
#include <stdlib.h>
#include <string.h>
#include <stdint.h>

static int display_info(const char *fpath, const struct stat *sb,
int tflag, struct FTW *ftwbuf)
{
    printf("%-3s %2d %7jd   %-50s %d %s\n",
        (tflag == FTW_D) ?   "d"   : (tflag == FTW_DNR) ? "dnr" :
        (tflag == FTW_DP) ?  "dp"  : (tflag == FTW_F) ?
            (S_ISBLK(sb->st_mode) ? "f b" :
             S_ISCHR(sb->st_mode) ? "f c" :
             S_ISFIFO(sb->st_mode) ? "f p" :
             S_ISREG(sb->st_mode) ? "f r" :
             S_ISSOCK(sb->st_mode) ? "f s" : "f ?") :
        (tflag == FTW_NS) ?  "ns"  : (tflag == FTW_SL) ?  "sl" :
        (tflag == FTW_SLN) ? "sln" : "?",
        ftwbuf->level, (intmax_t) sb->st_size,
        fpath, ftwbuf->base, fpath + ftwbuf->base);
    return 0;
}
```

- A large printf with nested if statements
- what do the flags and constants mean? Look for the documentation
- What are the arguments? What is the `stat` struct?
- What is ftw.h for?

Programming exercise 4 (cyclic linked lists) $(3+3+1+1=8$ points)

1. Implement cyclic linked lists according to the `C` header file below. Watch for memory leaks and use the `clist_tests.c` file for testing.
2. `clist_node_t` stores one integer per list node. What if other data should be stored in the list nodes, for example, two floating point numbers, or a custom `struct`? Outline three different approaches and briefly discuss their strengths and weaknesses.
3. This interface includes a function `clist_remove_after`, but no function `clist_remove`. Why would the latter function be inefficient?
4. This interface uses "double" pointers (`clist_node_t **`) to be able to modify the pointers passed in. For example,

   ```
   clist_node_t * clist = NULL;
   clist_insert_after(&clist, 1);
   clist_remove_after(&clist);
   ```

   will cause `clist` to be `NULL` after execution of the code block. This design decision is problematic: Replacing the last line with `clist_remove_after(&(clist->next))` results in undefined behavior. Why?

- Remember KDI. Make the next pointer of the tail point to the header.
- you only need to insert and remove directly after the head (i.e. at one place)
- Use the address sanitizer.

```
/*
 * Circular singly linked list
 *
 * Every node contains a pointer to the next node and data.
 * The nodes are ordered in a circular fashion, e.g.,
 *
 *   .-> 7 -> 2 -> 1 -> 33 -.
 *   |                      |
 *   .----------------------.
 *
 * A list is a pointer to a node, or NULL if the list is empty.
 *
 */

typedef struct CyclicListNode {
    struct CyclicListNode * next;
    int data;
} clist_node_t;

// inserts new node after clist; returns 0 if successful
int clist_insert_after(clist_node_t ** clist, int data);

// removes the node after clist; returns 0 if successful
// if the list becomes empty, sets clist to NULL
int clist_remove_after(clist_node_t ** clist);

// frees all nodes in clist, and sets clist to NULL
int clist_destroy(clist_node_t ** clist);
```

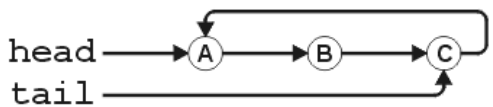


Figure 1. A cyclic linked list and three nodes.

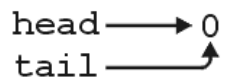


Figure 2. An empty cyclic linked list.

References I

[1] A. S. Tanenbaum und H. Bos, *Modern operating systems*. Pearson, 2015.
[2] W. Stallings, *Operating systems: internals and design principles*. Upper Saddle River, NJ: Pearson/Prentice Hall, 2009.
[3] A. Silberschatz, P. B. Galvin und G. Gagne, *Operating system principles*. John Wiley & Sons, 2006.
[4] B. W. Kernighan und D. M. Ritchie, *The C programming language*. 2006.