# 2021 exam set operatings systems

Operativsystemer (Danmarks Tekniske Universitet)

## Technical University of Denmark

Written Examination, December 9, 2021

Course: Operating Systems

Course No.: 02159

Aids allowed: All aids

Exam duration: 4 hours

Weighting: According to their respective points (100 points in total)

**Notes:**

- All questions contribute toward the final grade. The maximum number of points awarded by each correctly answered question is listed next to the question.
- For short questions, the following apply: (i) a brief answer is expected; (ii) you are also expected to briefly explain your answer in approximately 2-3 lines; (iii) a correct answer with a correct explanation yields 4 points in total.
- For long questions, the following apply: (i) there might be more than one correct answer; (ii) you are expected to briefly explain your answer in approximately half a page; (iii) a complete and correct answer yields 12 points; (iv) if you make additional assumptions, remember to briefly explain them.

### A. Short Questions

**Question A.1:** Name a communication method that can be used between threads of the same process but cannot be used for communication between a parent and a child process? Explain briefly your answer. (**4 points**)

**Question A.2:** What is the main advantage of preemptive scheduling compared to nonpreemptive scheduling? Explain briefly your answer. (**4 points**)

**Question A.3:** What will the code in Figure 1 print? Explain briefly your answer. You can assume that all system call invocations are successful. (**4 points**)

**Question A.4:** What will the code in Figure 2 most likely print? Explain briefly your answer. You can assume that all system call invocations are successful. (**4 points**)

**Question A.5:** What will the code in Figure 3 most likely print? Explain briefly your answer. You can assume that all system call invocations are successful. (**4 points**)

**Question A.6:** You are developing an application that is composed of multiple collaborative processes. You wish to implement the following functionality: if a resource is currently unavailable, the process should go to sleep until it receives a wakeup signal from another the process. Which method you would use to avoid a race condition? Explain briefly your answer. (**4 points**)

**Question A.7:** What is the key advantage of monolithic operating systems compared to microkernel operating systems? Explain briefly your answer. (**4 points**)

**Question A.8:** Which POSIX system call should you use to send the `SIGUSR1` signal to a child process? Explain briefly your answer. (**4 points**)

```c
#include <stdio.h>
#include <unistd.h>
#include <stdlib.h>

int counter = 1;

void child(void) {
    counter++;
    exit(0);
}

int main(void)
{
  if (fork() == 0) {
    child();
  }
  waitpid(-1, NULL, 0);

  printf("%d\n", counter);
  return 0;
}
```

Figure 1: Code of Question A.3.

```c
#include <stdio.h>
#include <pthread.h>

int A = 0;
pthread_t thread_id[1000];

void* count(void *input) {
  int i;
  for (i=0;i<1000;i++)
    A++;
  pthread_exit(NULL);
}

int main(void) {
  int i;
  for (i=0;i<1000;i++)
    pthread_create(&thread_id[i], NULL, count, NULL);
  for (i=0;i<1000;i++)
    pthread_join(thread_id[i], NULL);
  printf("%d\n", A);
  return 0;
}
```

Figure 2: Code of Question A.4.

```
#include <pthread.h>
#include <unistd.h>
#include <stdio.h>

int accounts[2];
pthread_mutex_t lock[2];
pthread_t tid1, tid2;

struct Transact {
  int from;
  int to;
  int amount;
};

struct Transact t1 = {.from = 0, .to = 1, .amount = 10};
struct Transact t2 = {.from = 1, .to = 0, .amount = 20};

void *transfer(void *in) {
  struct Transact trs = *(struct Transact*)in;

  pthread_mutex_lock(&lock[trs.from]);
  sleep(1);
  pthread_mutex_lock(&lock[trs.to]);

  accounts[trs.from]-=trs.amount;
  accounts[trs.to]+=trs.amount;

  pthread_mutex_unlock(&lock[trs.to]);
  pthread_mutex_unlock(&lock[trs.from]);
  pthread_exit(0);
}

int main(void) {
  pthread_mutex_init(&lock[0], NULL);
  pthread_mutex_init(&lock[1], NULL);
  accounts[0] = 100;
  accounts[1] = 100;

  pthread_create(&tid1, NULL, transfer, (void*)&t1);
  pthread_create(&tid2, NULL, transfer, (void*)&t2);
  pthread_join(tid1, NULL);
  pthread_join(tid2, NULL);
  printf("%d\n", accounts[0]);
}
```

Figure 3: Code of Question A.5.

```
#include <stdio.h>
#include <unistd.h>
#include <stdlib.h>

int main(void){
  fork();
  fork();
  printf("a\n");
  exit(0);
}
```

Figure 4: Code of Question A.9.

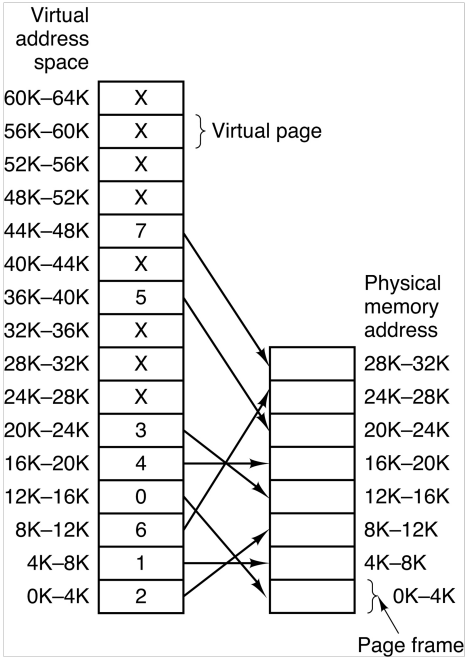


Figure 5: Data for Question A.12.

**Question A.9:** How many times will the letter 'a' be printed if we execute the code in Figure 4? Explain briefly your answer. You can assume that all system call invocations are successful. (**4 points**)

**Question A.10:** What is the primary role of the Memory Management Unit (MMU)? Explain briefly your answer. (**4 points**)

**Question A.11:** Which is the key disadvantage of the NFU (Not Frequently Used) page replacement algorithm? Explain briefly your answer. (**4 points**)

**Question A.12:** Assuming the current state of the memory is as shown in Figure 5, name a virtual address that will generate a page fault if accessed? Explain briefly your answer. (**4 points**)

**Question A.13:** The DMA chip can transfer data from and to the memory without using the CPU. Name a scenario that it is not efficient to use DMA for memory transfer? Explain briefly your answer. (**4 points**)

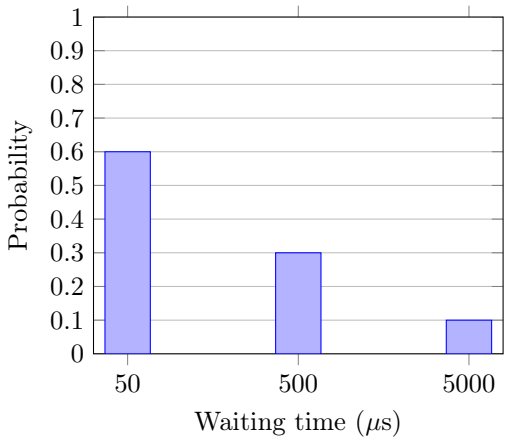


Figure 6: Data for Question B.2.

**Question A.14:** Which is the purpose of a watchdog timer? Explain briefly your answer. (**4 points**)

**Question A.15:** Which is the key advantage of developing device drivers using interrupt-driven I/O? Explain briefly your answer. (**4 points**)

**Question A.16:** In computer systems with multiple processors, name a challenge that characterises distributed systems as opposed to multicore processors? Explain briefly your answer. (**4 points**)

## B. Long Questions

### Question B.1 (12 points)

(i) Summarise the advantages and disadvantages of processes and threads.

(ii) You are developing a web browser. The web browser needs to support multiple parallel tabs, so that the user can browse multiple websites in parallel. Would you implement the browser using multiple processes (one process per tab) or multiple threads (one thread per tab)? Motivate your answer and discuss if the disadvantages of your solution (processes or threads) are relevant in this use case. If relevant, also discuss how you would overcome them.

### Question B.2 (12 points)

(i) Explain in your own words when using a spinlock (*i.e.*, a busy-waiting mutex) in a multiprocessor system can improve the performance.

(ii) You are using a multiprocessor system. The operating system implements hybrid mutexes. These mutexes operate as follows. If a thread requests to acquire a locked mutex, the thread first continuously polls the mutex (spins) for a period of time, $T$. After the time $T$ passes, the thread yields (context switch) and retries when it gets rescheduled. The parameter $T$ is configurable, taking values in $\mu$s in the range $[0, 65535]$. Setting $T = 0$ disables spinning.

After long-term statistical analysis, you know that the time a thread needs to wait for a locked mutex to be released follows the histogram provided in Figure 6. Moreover, a context switch takes 1000 $\mu$s.

Calculate the optimum value for the configuration parameter $T$, which minimises the overhead (*i.e.* the sum of time the CPU wastes spinning and switching). For simplicity, you can consider that when the thread gets rescheduled after the first context switch, it finds the mutex unlocked.

### Question B.3 (12 points)

After the presentation of the experiments of all other groups, propose a new experiment for the OS Challenge that your group has not tested before. Motivate the experiment by explaining why you believe it will improve the performance and describe how you would test if your hypothesis is true.