# Operating Systems Problem session 1

Dibran Dokter 1047390 & Marnix Lukasse 1047400

September 13, 2019

## 1.4

**Question:**
Describe the differences between symmetric and asymmetric multiprocessing. What are three advantages and one disadvantage of multiprocessor systems?

**Answer:**
In symmetric multiprocessing there is one "boss" processor that tells all the other processors what to do. In asymmetric multiprocessing all the processes are equal and choose their own work.

| **Advantages** | **Disadvantages** |
|---|---|
| Increased Throughput<br>Increased Reliability<br>Economy of Scale | More complex data sharing |

## 1.8

**Question:**
What is the purpose of interrupts? How does an interrupt differ from a trap? Can traps be generated intentionally by a user program? If so, for what purpose?
**Answer:**
The purpose of an interrupt is to signal to the processor that it needs to do something. Interrupt is hardware generated. A trap is a signal from a program that something went wrong (like division by 0) or a request to the operating system. Traps can be generated by user programs when the program needs the operating system to do something.

## 3.2

**Question:**
Describe the actions taken by a kernel to context-switch between processes.
**Answer:**
The system stores the current process state in its PCB and loads the PCB of the process to be executed. Then the loaded process is executed.

## 3.5

**Question:**
Including the initial parent process, how many processes are created by the program shown in Figure 3.31?
**Answer:**
5, 4 in the for loop and the parent process itself.

## 3.10

**Question:**
Using the program shown in Figure 3.34, explain what the output will be at lines X and Y .
**Answer:**
Line X:
"CHILD: 0 CHILD: -1 CHILD: -4 CHILD: -9 CHILD: -16 "
Line Y:
"PARENT: 0 PARENT: -1 PARENT: -4 PARENT: -9 PARENT: -16 "

## 3.11

**Question:**
What are the benefits and the disadvantages of each of the following?
Consider both the system level and the programmer level.
a. Synchronous and asynchronous communication
b. Automatic and explicit buffering
c. Send by copy and send by reference
d. Fixed-sized and variable-sized messages
**Answer:**

a. Synchronous communication allows you to be sure that the message has been received before continuing execution. It also requires the program to block on sending and receiving. This is easier to implement because there is no buffering. The big downside is one of the programs is blocked.

b. With automatic buffering the sender is never blocked because the queue is never full. When using a explicit buffer the queue has a bound that will block the sender when full. Explicit buffering requires less memory.

c. Send by copy does not allow the receiver to change the value, send by reference does. Sending by copy is safer because the data cannot suddenly change in the other process. Sending by copy however requires the system to copy the data which can be expensive with big amounts of data. Copying the data allows two instances of the same program with a different set of data. The programs can work with different data sets. When referencing you use less memory and should be used when both processes should use the same data.

d. With fixed-sized the system implementation is simple, with variable-size the implementation is more difficult. The programming with a variable-size message is easier than fixed-size.

## 4.5

**Question:**
In Chapter 3, we discussed Googles Chrome browser and its practice of opening each new website in a separate process. Would the same benefits have been achieved if instead Chrome had been designed to open each new website in a separate thread? Explain.
**Answer:**
No, it would have better performance because creating a thread has less overhead than creating a new process. But the biggest benefit is that when one process crashes the program does not. In the case of threads the program crashes. Another risk is that because the data is shared, when another thread crashes the data for the other threads can be tainted.

## 4.8

**Question:**
Determine if the following problems exhibit task or data parallelism:
• The multithreaded statistical program described in Exercise 4.16
• The multithreaded Sudoku validator described in Project 1 in this chapter
• The multithreaded sorting program described in Project 2 in this chapter
• The multithreaded web server described in Section 4.1
**Answer:**
• Task parallelism.
• Data parallelism.
• Data parallelism, we assume the merge thread starts after the first 2 threads have finished.
• Task parallelism.

## 4.10

**Question:**
Consider the following code segment:
pid t pid;
pid = fork();
if (pid == 0) { /* child process */

```
    fork();
    thread create( . . .);
}
fork();
```

a. How many unique processes are created?
b. How many unique threads are created?

**Answer:**

a. 6.

b. We assume the new processes duplicate all threads in the parent process. Then we get to 10 threads.