Report for Lab 1

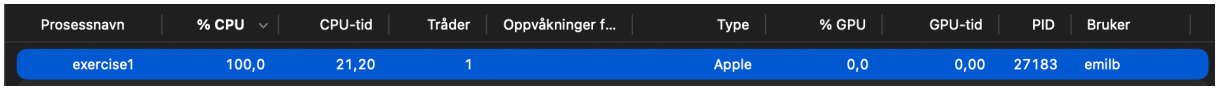
Name: Emil Berglund

GitHub: EmilB04

Disclaimer: I did figure 7-9 on my local machine, due to many people being on the hiof server. Thus there is different GUI and file-names

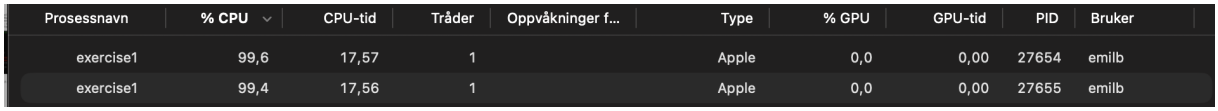
Exercise 1:

Figure 7: Making Cpu0 to run out



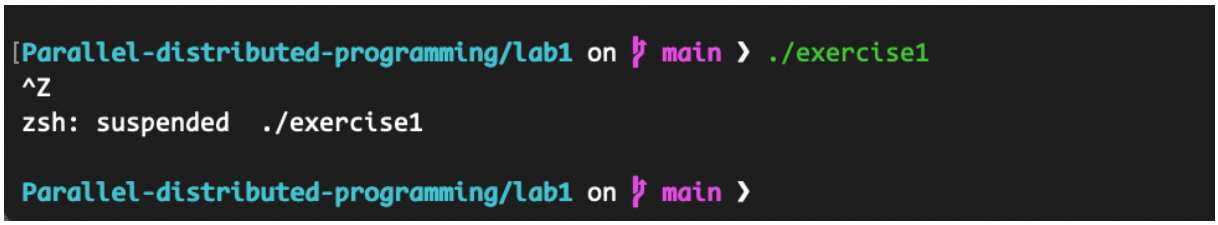
Prosessnavn	% CPU	CPU-tid	Tråder	Oppvåkninger f...	Type	% GPU	GPU-tid	PID	Bruker
exercise1	100,0	21,20	1		Apple	0,0	0,00	27183	emilb

Figure 8: Two CPUs are running



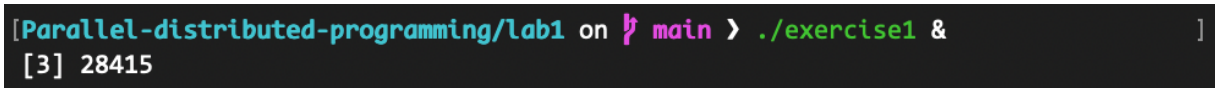
Prosessnavn	% CPU	CPU-tid	Tråder	Oppvåkninger f...	Type	% GPU	GPU-tid	PID	Bruker
exercise1	99,6	17,57	1		Apple	0,0	0,00	27654	emilb
exercise1	99,4	17,56	1		Apple	0,0	0,00	27655	emilb

Figure 9: Stop running a program with Ctrl- Z.



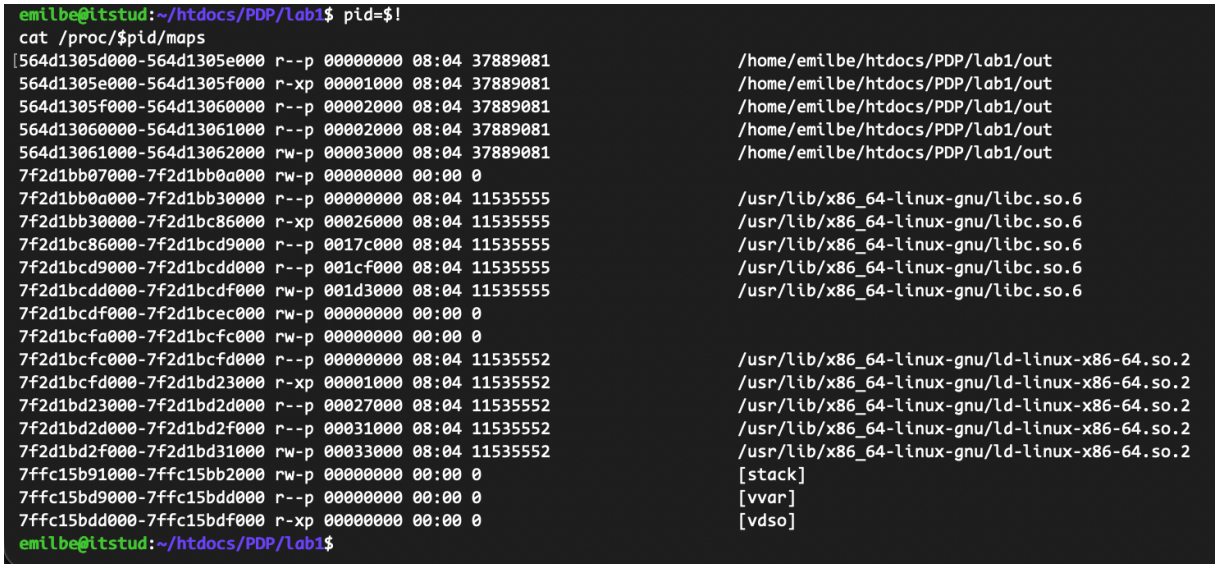
```
[Parallel-distributed-programming/lab1 on  main ❯ ./exercise1
 ^Z
 zsh: suspended  ./exercise1

 Parallel-distributed-programming/lab1 on  main ❯
```



```
[Parallel-distributed-programming/lab1 on  main ❯ ./exercise1 &                ]
 [3] 28415
```

Figure 10: Memory address of the Hello World program



```
 emilbe@itstud:~/htdocs/PDP/lab1$ pid=$!
 cat /proc/$pid/maps
[564d1305d000-564d1305e000 r--p 00000000 08:04 37889081                  /home/emilbe/htdocs/PDP/lab1/out
 564d1305e000-564d1305f000 r-xp 00001000 08:04 37889081                  /home/emilbe/htdocs/PDP/lab1/out
 564d1305f000-564d13060000 r--p 00002000 08:04 37889081                  /home/emilbe/htdocs/PDP/lab1/out
 564d13060000-564d13061000 r--p 00002000 08:04 37889081                  /home/emilbe/htdocs/PDP/lab1/out
 564d13061000-564d13062000 rw-p 00003000 08:04 37889081                  /home/emilbe/htdocs/PDP/lab1/out
 7f2d1bb07000-7f2d1bb0a000 rw-p 00000000 00:00 0
 7f2d1bb0a000-7f2d1bb30000 r--p 00000000 08:04 11535555                  /usr/lib/x86_64-linux-gnu/libc.so.6
 7f2d1bb30000-7f2d1bc86000 r-xp 00026000 08:04 11535555                  /usr/lib/x86_64-linux-gnu/libc.so.6
 7f2d1bc86000-7f2d1bcd9000 r--p 0017c000 08:04 11535555                  /usr/lib/x86_64-linux-gnu/libc.so.6
 7f2d1bcd9000-7f2d1bcdd000 r--p 001cf000 08:04 11535555                  /usr/lib/x86_64-linux-gnu/libc.so.6
 7f2d1bcdd000-7f2d1bcdf000 rw-p 001d3000 08:04 11535555                  /usr/lib/x86_64-linux-gnu/libc.so.6
 7f2d1bcdf000-7f2d1bcec000 rw-p 00000000 00:00 0
 7f2d1bcfa000-7f2d1bcfc000 rw-p 00000000 00:00 0
 7f2d1bcfc000-7f2d1bcfd000 r--p 00000000 08:04 11535552                  /usr/lib/x86_64-linux-gnu/ld-linux-x86-64.so.2
 7f2d1bcfd000-7f2d1bd23000 r-xp 00001000 08:04 11535552                  /usr/lib/x86_64-linux-gnu/ld-linux-x86-64.so.2
 7f2d1bd23000-7f2d1bd2d000 r--p 00027000 08:04 11535552                  /usr/lib/x86_64-linux-gnu/ld-linux-x86-64.so.2
 7f2d1bd2d000-7f2d1bd2f000 r--p 00031000 08:04 11535552                  /usr/lib/x86_64-linux-gnu/ld-linux-x86-64.so.2
 7f2d1bd2f000-7f2d1bd31000 rw-p 00033000 08:04 11535552                  /usr/lib/x86_64-linux-gnu/ld-linux-x86-64.so.2
 7ffc15b91000-7ffc15bb2000 rw-p 00000000 00:00 0                         [stack]
 7ffc15bd9000-7ffc15bdd000 r--p 00000000 00:00 0                         [vvar]
 7ffc15bdd000-7ffc15bdf000 r-xp 00000000 00:00 0                         [vdso]
 emilbe@itstud:~/htdocs/PDP/lab1$
```

Exercise 2

Figure 15: One possible output of top command.

```
Hello World from Thread #8!
Another message from Thread #8!

Hello World from Thread #9!
Another message from Thread #9!

top - 14:49:15 up 47 days, 22:05, 19 users,  load average: 7.06, 7.26, 7.58
Tasks: 288 total,   8 running, 248 sleeping,  29 stopped,   3 zombie
%Cpu0  :100.0 us,  0.0 sy,  0.0 ni,  0.0 id,  0.0 wa,  0.0 hi,  0.0 si,  0.0 st
%Cpu1  :100.0 us,  0.0 sy,  0.0 ni,  0.0 id,  0.0 wa,  0.0 hi,  0.0 si,  0.0 st
MiB Mem :   3915.2 total,   1155.0 free,   1008.7 used,   2059.8 buff/cache
MiB Swap:   7629.0 total,   7446.2 free,    182.8 used.   2906.5 avail Mem

    PID USER      PR  NI    VIRT    RES    SHR S  %CPU  %MEM     TIME+ COMMAND
3941724 heddaon   20   0    2336    880    792 R  28.8   0.0  18:56.99 out
3942323 mariuror  20   0    2336    896    808 R  28.8   0.0  19:32.60 out
3945404 mrgarabe  20   0    2336    888    800 R  28.8   0.0   8:57.91 out
3681392 ttdinh    20   0    2336    892    804 R  28.5   0.0     74,42 out
3941475 aleksj    20   0    2336    924    832 R  28.5   0.0  22:15.59 out
3940826 simenko   20   0    2336    884    792 R  27.8   0.0  25:01.37 HW
3943095 sebastwt  20   0    2336    876    784 R  27.8   0.0  17:54.57 out
3944216 mrgarabe  20   0   13592   6280   3984 S   0.3   0.2   0:08.39 top
3949371 emilbe    20   0   13748   6412   4008 R   0.3   0.2   0:00.25 top
```

- Due to high load on the server, 2/2 CPUs are running
- Since I only defined 100 threads, I got 100 threads:

```
Hello World from Thread #99!
Another message from Thread #99!
```

Exercise 3:

Disclaimer. Since it's relatively long ago since I programmed in C, I got some help from ChatGPT5 with this exercise, mainly the code/syntax, and how to allocate memory.

Screenshot and explanation below.

```
  GNU nano 7.2                                  ram_usage.c
#include <stdio.h>
#include <stdlib.h>
#include <unistd.h>

int main(void) {
  const size_t MB = 10u * 1024u * 1024u;  // 10 MB per step
  size_t total = 0;

  while (1) {
    void *p = malloc(MB);  // allocate 1 MB
    if (!p) {
      fprintf(stderr, "Allocation failed after ~%zu MB\n", total / (1024u*1024u));
      return 1;
    }

    // Touch the memory so the OS backs it with RAM
    for (size_t i = 0; i < MB; i++) {
      ((char *)p)[i] = 0;
    }

    total += MB;
    printf("Allocated ~%zu MB total\n", total / (1024u*1024u));

    sleep(1);  // pause 1 second
  }
}
```

Includes:

- Stdio.h for printing with printf
- Stdlib.h for malloc and size_t
- Unistd.h for sleep

Constants and counters:

- MB is set to 10 x 1024 x 1024 bytes (that's 10 MB).
- Total keeps track of how many bytes that has allocated so far.

Infinite loop:

- Malloc(MB): Asks the OS for a new 10MB block of memory
    - If malloc returns NULL, there wasn't enough memory, and the program will show the total amount of memory used and exit
- Touch the memory:
    - The for-loop writes 0 into every byte of the new block
    - This is important because many operating systems don't give you real RAM until you actually use the memory. Writing to it forces the OS to back those virtual pages with physical memory, so the program's RAM usage truly increases.

- Update and print:
    - Add 10 MB to total
    - Print the total allocated so far in megabytes
- Sleep(1):
    - Wait for 1 second before the next iteration. This makes the growth gradual and you can easily see the update

No free:

- We never free the blocks on purpose, so the program's memory usage keeps growing by 10 MB each second until allocation fails or you stop the program (Ctrl-C).

Exercise 4:

Running the code:

-
```
Hello World from Thread #599!
Another message from Thread #599!
```

Using top to see which CPU is running the code:

- Since the program is so fast, it is not possible to see which CPU is being used, when running the program. In other words, it does not show up in the overview when writing "top".

Explaining why Pthread programming is not parallism:

- Pthreads are a tool for concurrency while parallelism is a hardware and runtime property. Threads have no guaranteed parallel execution. If you have only one CPU core, the OS time-slices threads; meaning they run one at a time.
- The code in this exercise demonstrates concurrency (many threads created), but not meaningful parallelism. This is because they mainly print, which is I/O-bound and short lived.