Exercise session
(Processes)

Operating Systems – EDA093/DIT401

UNIVERSITY OF
GOTHENBURG

Question 1

Describe the actions taken by a kernel to context-switch between processes

See slides for lecture 2...

Question 2

What is printed by this program?

CHILD 0
CHILD -1
CHILD -4
CHILD -9
CHILD -16
PARENT 0
PARENT 1
PARENT 2
PARENT 3
PARENT 4

```c
#include <stdio.h>
#include <sys/types.h>
#include <unistd.h>

#define SIZE 5

int nums[SIZE] = {0,1,2,3,4};

int main()
{
    int i;
    pid_t pid;
    pid = fork();
    if (pid == 0) {
        for (i = 0; i < SIZE; i++) {
            nums[i] *= -i;
            printf("CHILD %d\n",nums[i]);
        }
    }
    else if (pid > 0) {
        wait(NULL);
        for (i = 0; i < SIZE; i++)
            printf("PARENT: %d\n",nums[i]);
    }
    return 0;
}
```

Question 3

What is printed by this program?

child: pid = 0
child: pid1 = X
parent: pid = X
parent: pid1 = Y

Notice:
X is the same for child and parent
Y is different from X
X > 1
Y > 1
The parent is not waiting for the child to print. Messages can be printed in different order...

```c
#include <stdio.h>
#include <sys/types.h>
#include <unistd.h>

int main()
{
    pid_t pid, pid1;
    pid = fork();
    if (pid < 0) {
        fprintf(stderr, "Fork Failed");
    }
    else if (pid == 0) {
        pid1 = getpid();
        printf("child: pid = %d", pid)
        printf("child: pid1 = %d", pid1)
    }
    else if (pid > 0) {
        pid1 = getpid();
        printf("parent: pid = %d", pid)
        printf("parent: pid1 = %d", pid1)
        wait(NULL);
    }
    return 0;
}
```

Question 4

Consider a multiprogrammed system with degree of 5. If each process spends 40% of its time waiting for I/O, what will be the CPU utilization?

CPU utilization = $1 - p^n = 1 - 0.4^5 = 0.99$

where p is the probability for a process to be waiting for I/O.

Question 5

To use cache memory, the main memory is divided into cache lines, typically 32 or 64 bytes long. An entire cache line is cached at once. What is the advantage of caching an entire line instead of a single byte or word at a time?

Question 5 - Solution

- Locality of reference: if one location is read then the probability of accessing nearby locations next is very high, particularly the following memory locations. So, by caching an entire cache line, the probability of a cache hit next is increased.
- Modern hardware can do a block transfer of 32 or 64 bytes into a cache line much faster than reading the same data as individual words.