King Saud University
Department of Computer Science
CSC227:Operating Systems
Tutorial No. 3

Q 1) What is a process?
Executable program with its own address space and process control block.

Q 2) How does a child process differ from its parent?
c. It uses separate address space and separate stack pointer from the parent pointer.

Q 3) What is a process control block (PCB)?
Answer:
A process control block is a data structure used by an operating system to manage processes.

Q 4) Is a PCB found all operating systems?
Answer:
Process control blocks are found in all modern operating systems.

Q 5) Is PCB in all operating systems the same?
The structure of PCBs will be different on each operating system though.

Q 6) What does process context switching involves?
Answer:
Changing Process Control Block structures between an old and a new process

Q 7) Context switching operation represents a pure overhead in the OS process management. What does it means?
Answer:
It means that no other useful work can be done during the context switch operation.

Q 8) List at least FIVE other kinds of data that will be stored in a PCB that is not shown in PCB diagram.
Answer:
PCB contains a great deal of information about the process, including:
- The scheduling priority
- Information about the user and groups associated with this process
- Pointer to the parent process PCB
- Pointer to a list of child process PCBs
- Pointers to the process's data and machine code in memory
- Pointers to open files and other resources held by the process

Q 9) Describe the actions a kernel takes to context switch between processes.
Answer:
In general, the operating system must save the state of the currently running process and restore the state of the process scheduled to be run next. Saving the state of a process typically includes the values of all the CPU registers in addition to memory allocation.

Context switches must also perform many architecture-specific operations, including flushing data and instruction caches.

Q 10) A computer has multiple register sets. Describe the actions of a context switch if the new context is already loaded into one of the register sets. What else must happen if the new context is in memory rather than a register set, and all the register sets are in use?

Answer:

The CPU current-register-set pointer is changed to point to the set containing the new context, which takes very little time. If the context is in memory, one of the contexts in a register set must be chosen and moved to memory, and the new context must be loaded from memory into the set. This process takes a little more time than on systems with one set of registers, depending on how a replacement victim is selected.

Q 11) What is the advantage of restricting a child process to a subset of the parent's process?

Answer: the processes will limit within the resource allocated to parent and thus will not exhaust system resources.

Q 12) List the reasons when parent may terminate the execution of its child.

Answer: the child has exceeded its usage of system resources, the task assigned to the child is no longer required, the parent is exiting and the OS does not allow a child to continue if its parent terminates.)

Q 13) How does provision of multiple registers help in context switch?

Answer: if multiple registers are provided then the context switch simply requires changing the pointer to the current register set. Of course, if there are more process than the number of registers available, the OS must copy the register data to and form memory.

Q 14) Mention the three different queues a process may be place in by the OS before it completes execution.

Answer: job, ready, waiting (device)

Q 15) In a context switch, the OS made changes to some fields in the PCB of the current running process. Mention two fields in the PCB that will be updated with new information.

Answer: register, state

Q 16) How many times the message will be printed?

```
main()
{
        fork();
        printf("Hello world");
}
```

Answer.: 2 times. Child is created at fork() and every line after fork will be executed twice once by parent (i.e. printf("Hello world");) and once by child (again printf("Hello world");)

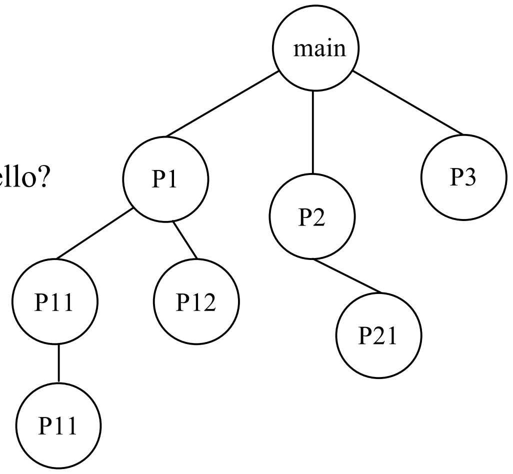
Q 17) Consider the following C language program, what are the possible outputs?

```
#include <stdio.h>
int num;
int main(){
   num = 1;
   fork();
   num = num + 1;
   printf(num);
}
```

Answer: It will print value of num two times one from parent and one from child.

Q 18) How many times does the program below print Hello?

```
include <stdio.h>
int main()
{
     fork();
     fork();
     fork();
     printf( "Hello\n" );
}
```

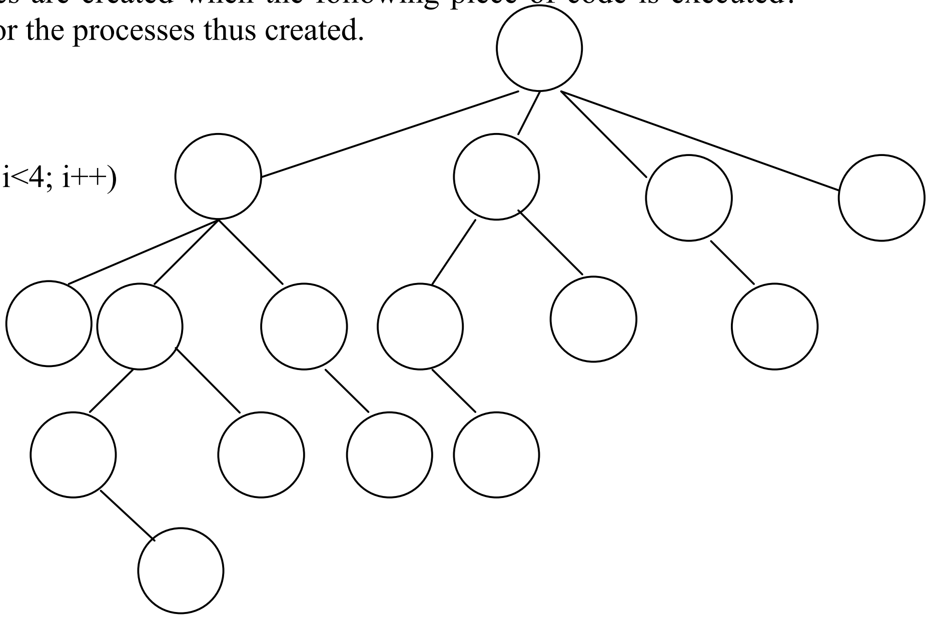


Answer: 8 times.

Q 19) How many processes are created when the following piece of code is executed? Draw the process tree for the processes thus created.

```
int main() {
     int i;

     for (i=0; i<4; i++)
     fork();
     return 1;
}
```



Answer: 15 + main

The diagram is not accurate please consider the notes from the tutorial lecture