This is a self-evaluation test. All the questions are related to previous concepts which are assumed to be known by you. The questions are oriented to Linux systems and i386 architectures. We will give you the solution briefly. The questions have been grouped in topics: O.S., architecture and C programming.


If we execute the following system call in a C program

\[ \text{Write(1,"HELLO\n",10);} \]
- We will write until \n
- We will write 10 bytes
- I don’t know

*The clock interruption code is normally executed
- In user mode
- In system mode
- I don’t know

Where and when the OS processes are normally planned?
- Every x tics of the clock interruption
- Every tic of the clock interruption
- When a fork is done
- I don’t know

What is the idea of “devices independence”?
- Decouple the API and the input/output functionality seen by the processes of the specific features of the devices
- Differentiate the devices between them
- To avoid having a process bothering another process when accessing to the devices
- I don’t know

In the priorities policies
- If we don’t re-calculate some priorities some process could be queued indefinitely.
- It is processes execution is guaranteed even if there is not any re-calculation.
- I don’t know

The syntax of the system calls related to the input/output
- Depends of the device
- Does not depend of the device
- I don’t know

What is a PCB?
- It is the data structure which defines a process
- It is the data structure which manages the active processes of the system
- I don’t know

What is a process in a READY state?
- It is a process which is being prepared to be selected by the planner
- It is a process which is waiting an input/output
- It is a process prepared to be executed but is not executed because it has low priority
- I don’t know
*What is a ZOMBIE process?*

() A process that has finished its execution due to an error
() A process that has finished its execution due to an exception
() A process that has finished its execution and its finish state has not been checked by its father yet
() I don’t know

*What is the processes planner?*

() It is the system code which decides if it is necessary to change the context and apply the planning policy
() It is the code which decides when an application finishes
() I don’t know

*What is the idle process?*

() It is a user process which generates a clock interruption every second to check if there is any process with more priority and change the context.
() It is a process or thread of the system which has the lowest priority.
() It is a low priority user process
() I don’t know

*The code of the system libraries…*

() Is executed in user mode
() Is executed in system mode
() Depends on the system call
() I don’t know

*Normally, the OS code to manage a system call*

() Has a common entry point
() Has an entry point for each system call
() I don’t know

*What is a “quantum”?*

() It is the time assigned to a process before getting a context change
() It is the time elapsed between two clock interruptions
() I don’t know

*What is a “clock tic”*

() It is the time assigned to a process before getting a context change
() It is the time elapsed between two clock interruptions
() I don’t know

*If we have virtual memory as the memory management system*

() We need all the pages of a process to be loaded
() We will have the needed pages loaded in memory
() We will only have 1 code page, 1 data page and 1 page for the stack
() I don’t know

*By default*

() The channels 0 and 1 are (respectively) the standard input and output
() The channels 0 and 1 are (respectively) the standard output and input
() I don’t know

*The write system call returns zero*

() When an error occurs
() When it has written 0 bytes
() When it is at the end of the file
() Never
() I don’t know
If we want to write a buffer of characters, the most efficient option is
   () A loop which executes 1 write of 1 byte every iteration
   () A write passing the starting address of the buffer
   () I don’t know

In a user program, how can we know the exact error produced by a system call?
   () Checking the return value of the call
   () Checking the errno variable
   () The system writes a message in the screen
   () I don’t know

How can we write in the screen from the system code?
   () With a printf
   () With a write
   () With a printk
   () I don’t know

A write system call implies…
   () Changing from real mode to protected mode
   () Changing from user mode to system mode
   () It does not imply any mode change
   () I don’t know

If a process does a fork of another process
   () Both processes share the code and the data
   () Both processes share the data but the code is not copied
   () Both process share the code but the data is not copied
   () I don’t know

EC2: Architecture

What is the difference in the esp register between these two instructions?
   Addl $4,%esp  or  popl %eax
   () No difference
   () The first one modifies esp and the second one not
   () The first one adds 4 to the esp and the second one substracks 4 to the esp
   () I don’t know

When a character is passed as a parameter in a function, the compiler reserves in the
stack space for
   () 1 byte
   () 2 bytes
   () 4 bytes
   () I don’t know

Which are the only registers saved and restored by the compiler if a subroutine
modifies them?
   () ebx,esi,edi
   () eax,ecx, edx
   () ds,es,sp
   () I don’t know
*When the compiler generates code for a subroutine, which registers does it assume that will be saved by the calling code if used?
   () ebx,esi,edi
   () eax,ecx,edx
   () ds,es,sp
   () I don’t know

*The result of a subroutine is returned
   () In the stack
   () In the eax register
   () I don’t know

*When an interruption is generated
   () We change to protected mode firstly and then we execute the system code
   () We go to the system code firstly and we change to protected mode before the first instruction
   () Depends of the interruption
   () Changing to protected mode and going to the system code is done atomically
   () I don’t know

*What are segment registers?
   () The registers pointing out to the system stack
   () The registers which define the memory segments
   () Privileged registers which can only be modified from system mode
   () I don’t know

*What is pagination?
   () A method to manage the memory
   () A method to manage the disk
   () I don’t know

*To make a stack change, without changing the execution or process mode
   () ss and esp have to be changed always
   () changing just one of them is enough
   () I don’t know

*What is the difference between ret and iret instructions?
   () There is no difference, they are equivalent
   () ret returns from a subroutine and iret returns from a system input
   () iret returns from a subroutine and ret returns from a system input
   () I don’t know

*What is the purpose of the EOI?
   () To indicate the finishing of a system call
   () To uninhibit the interruptions
   () To indicate to the interruptions controller that an interruption has been already treated
   () I don’t know

C PROGRAMMING / MAKEFILES

*What does the –I option of the C compiler do?
   () Defines the directory where the include files are located
   () Defines the name of the include files which have to be used
   () I don’t know
*What does the –O2 option of the C compiler do?
  () Defines the optimization level to 2
  () Defines the name of the compilation output
  () I don’t know

*If we define the following variable in a C program
  char *buffer;
  () We have defined a vector of an infinite size
  () We have defined a not initialized pointer to character
  () We define a pointer to character and the compiler is in charge of reserving space for it
  () I don’t know

*If A and B are declared as “int * A,*B;”, what happens if the make the following assignation
  *A=&B;
  () We assign to the memory position pointed out by A the address of the B variable
  () We assign to A the address of B
  () We assign to A the contents pointed out by B
  () I don’t know

*If we declare the A variable as “char *A”, and we immediately make read(0,A,100)
  () We read 100 bytes from the channel 0 and we save it in the address pointed out by A
  () We read 100 bytes from the channel 0 and we save it in the address pointed out by A. It could happen anything because A is not initialized.
  () We read 100 bytes from the channel 0 and we save it in the address pointed out by A. The compiler will reserve space in order to avoid errors because A is not initialized.
  () I don’t know

*If we execute the following function call in a C program:
  F(a,b,c)
  () The parameters are put in the stack from right to left
  () The parameters are put in the stack from left to right
  () The parameters are put in registers
  () I don’t know

*In a C program, the vectors and matrix are passed to the functions
  () By value
  () By reference
  () Depends on the size
  () I don’t know

*Who is the sender of a “Undefined symbol” message?
  () The compiler
  () The linker
  () The makefile
  () I don’t know
*When we get a warning indicating that we have indicated the prototype of a function
   () Is because we haven’t linked with the correct files
   () Is because we haven’t defined the header but it doesn’t matter because the
     compilers defines it well
   () Is because the function does not have the header defined
   () I don’t know
*If s is a pointer to a struct which contains a c field, how can we access this field?
   () (*s)->c
   () s->c
   () s.c
   () I don’t know