Midterm

Q2

1. What is the name of the system call that will allow the parent to detect the termination of its child?

Wait()

1. What process is responsible with the creation of other processes in the system?  
   The init process
2. Describe the exec(..) command; What are the meaning of its arguments.  
   The exec() command is to overwrite its address space and execute different code than parent. The meaning of its arguments is the name and location of an executable file.
3. What the permission: drw- rw- r—means?  
   drw-rw-r-- : represents the permission settings for the owner, group, and others.

D: the directory entry  
rw- : the owner can read and write the file  
rw- : the group can read and write the file  
r-- : others can only read the file

1. On which Unix directory most of the executable files are stored?  
   Most of the executable files are stored on **the binary directory.**

Q3  
a) A process that does “busy waiting” is said to be what state?  
 The running state

b) What condition to CS problem is not satisfied by the Mutual Exclusion with TS(lock) implementation? Explain why is not satisfied?

The multiprocessor environment to CS problem is not satisfied by the mutual exclusion with TS (lock) implementation. It does not matter enabling or disabling an interrupt on one CPU because it does not affect the other CPU. Also, disabling interrupts on a multiprocessor can be time consuming, since the message is passed to all the processors. This message passing delays entry into each critical section, and system efficiency decreases.

c) If TS (lock) is NOT executed atomically an additional condition to CS is not satisfied. Name it and give the specific execution sequence that shows it. Mention where exactly the interrupt inside the TS occurs.

The mutual exclusion condition to CS is not satisfied if the TS (lock) is not executed atomically.

While(true){

…….

……

Disable interrupts

CS

Enable interrupts;

}

Q5:

1. No, it is not possible for a child who has a winning ticket to starve. It is because each child has his ticket[i] that contains a unique number; when the clown randomly picks up a winning ticket, and if the number on winning ticket is equal to the number on the child’s ticket, the child will be called and get the money.

C1: ticket[1]  
C2: ticket[2]  
C3: ….

Clown:  
if(number on winning ticket == number on ticket[1]) {  
 called[1] = true;  
 give the prize;  
 totalAmount - = 20;

}

Child:

C1: While(! Called[1]) {exits}; // because called[1] is true  
 get money();  
 go hone();

1. Yes, because the child has the access to the shared variable totalAmount.
2. No, we assume that all winners have been called (called[j] = True), and this means that each child has the ticket containing a unique number. When the clown pick one winning ticket, each child should be called. Hence, it is not possible for the winning number never leave (winner < 5 never false).
3. Yes, it is possible because this is a high level language. The computer must first load the totalAmount, and the totalAmount will be in increament by 1 and then stored. However, another thread can load totalAmount before the first thread stores it. There is possibility for data coherence.  
     
   C1: totalAmount // load 0  
   C1: totalAmount+= 10 // inc 10  
   C2: totalAmount // load 10  
   C2: totalAmount+= 10 // inc 10  
   C1: totalAmount = 10 // store  
   C2: totalAmount = 10 // store