Real Time Systems and Control Applications

Contents

Midterm 1 Solution

1. Classification of real time systems (e.g. hard, firm or soft real time systems) depends on how to define cost of missing deadline. (T)

2. A good algorithm for scheduling hard real-time tasks must try to complete each task in the shortest possible time. (F)

3. Soft real-time tasks are those which do not have any time bounds associated with them. (F)

4. Function module_init() is invoked by insmod to initiate module's functions. You can use this function to allocate system resources, declare and start tasks. (F)

5. Using threads to implement multiple tasks is more efficient than using processes. One of the reasons is that it is much quicker to create a thread in a process. (T)

6. Unlike time sharing systems, the principal objective of batch processing systems is to minimize response time. (F)

7. The hardware allows privileged instructions to be executed only in kernel mode. (T)

8. A drawback of cyclic executive scheduler is that the decision table can be unlimited in length to handle long-lasting periodic tasks. (F)

9. Since current version of Linux has real time extensions, it is no longer necessary for a privileged user to use an RTOS to implement time systems. (F)

10.Code written for the Linux kernel space should not have a main() function. (T)

Given the following three tasks: T1 (0; 10; 3; 10); T2 (10; 3; 10); T3 (10; 3), which of the following is correct?

- a) They all represent the same task.
- b) T1 is a special case of T2, and T2 is a special case of T3.
- c) T2 represents a task which has an arbitrary phase.
- d) All of the above.

What Is The Output?

Race Condition!
Value of g cannot be determined.

```
#include <stdio.h>
#include <pthread.h>
 int g = 0;
 void *aThread()
{
    g++;
    pthread_exit(NULL);
}

int main (int argc, char *argv[])
{
    int i;
    pthread_t thread[3];
    for (i=0; i<3; i++)   {
        if( pthread_create( thread+i, NULL, aThread, NULL) )    {
            printf("ERROR; return code from pthread_create()\n");
            return -1;
        }
    }
    printf("The value of g is %d\n", g);
     return 0;
}
```

```
main(int argc, char ** argv)
{
    int child = fork();
    int c = 5;
    if(child == 0)
        c += 5;
    else  {
        child = fork();
        c += 10;
        if(child)
            c += 5;
    }
}
```

How many copies of c?

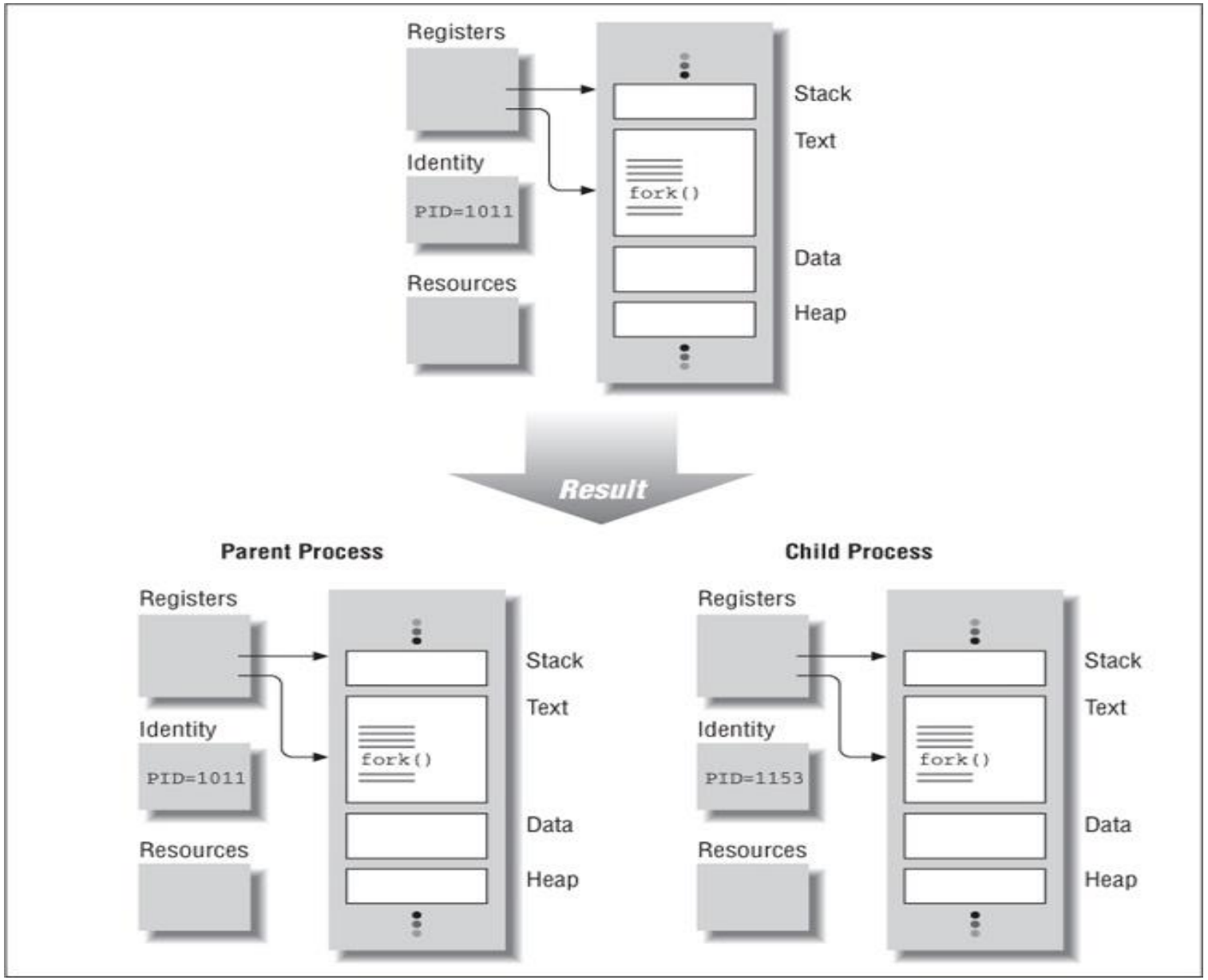
What are the values?

3 copies (Parent: 20, Child1:10, Child2: 15)

```
int main() {
    pid_t pid;
    char * str;
    str = malloc( 100 );
    strcpy( str, "Hello" );
    pid = fork();
    if( pid == 0 ) {
        printf( "In Child, the string is: %s\n", str );
        strcpy( str, "Goodbye" );
        printf( "In Child, the string is: %s\n", str );
        free( str );
    }
 else if( pid>0) {
        printf( "In Parent array, the string is: %s\n", str );
        sleep(1);
        printf( "In Parent array, the string is: %s\n", str );
        free( str );
    }
    else
        printf( "Error with fork()\n" );

    return 0;
}
```

```
In Parent array, the string is Hello
In Child, the string is Hello
In Child, the string is Goodbye
In Parent array, the string is Hello
```



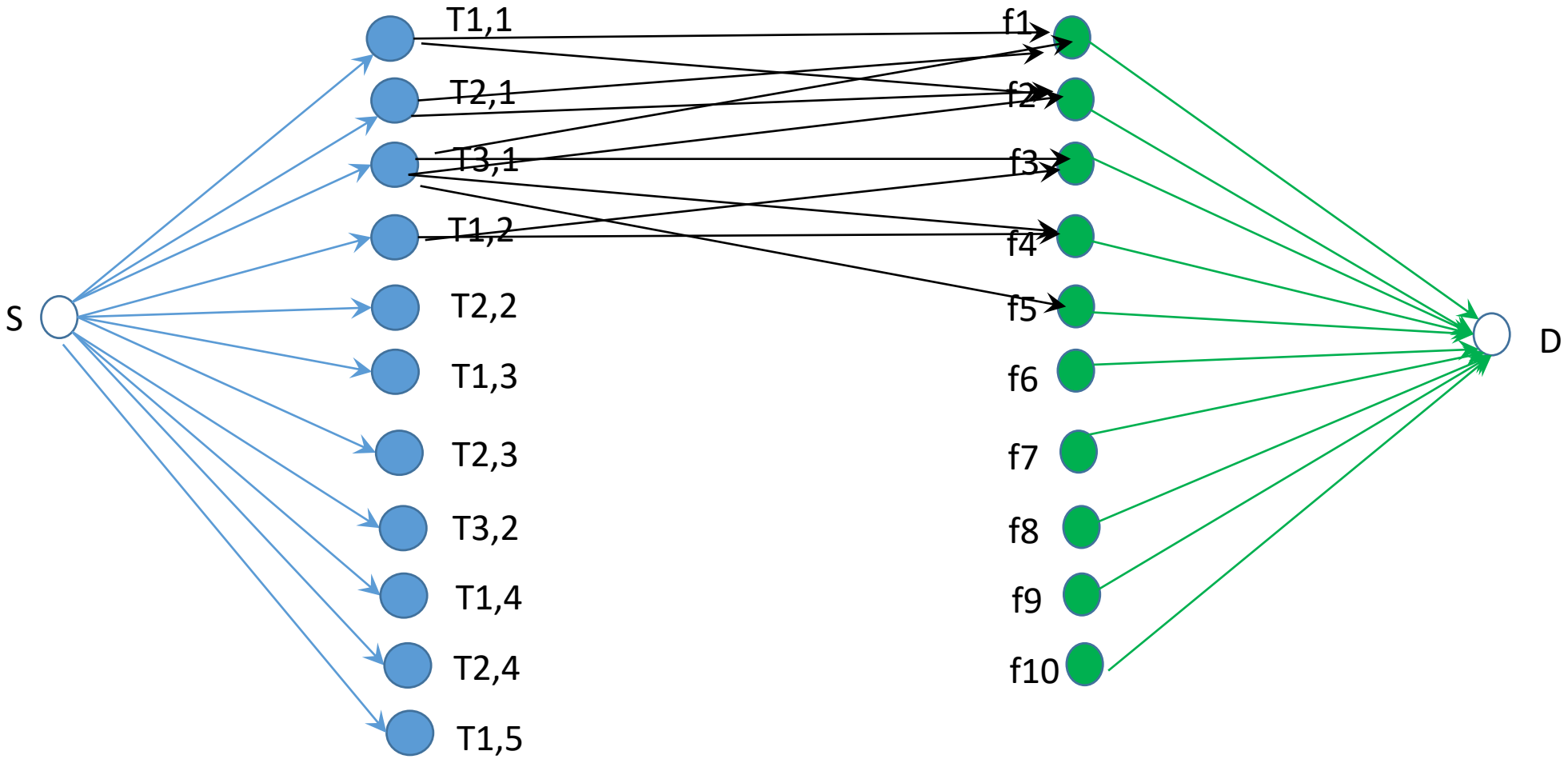
- When fork() is invoked, everything will be copied to a separated address space.
- The change of variable values in child process will not affect the parent process.

Short Answer

- Given the task set: T1(4, 1); T2(5, 1); T3(10, 2).
- Hyperperiod: 20
- Frame size: f=2
- Schedule: T1, T2, T3, T3, T1, Idle, T2, Idle, T1, Idle,
  T3, T3, T1, T2, Idle, Idle, T1, T2, , Idle, Idle.

（Note that the first idle cannot be T2）

Use network flow to model the CE scheduling



Use network flow to model the CE scheduling