Metropolia
Hannu Laine

Teamwork TI09S1E
15.3.2013

Topic for Teamwork project of "Real time programming"

Topic

In this topic you use a so called multi sensor simulator. Multi sensor simulator simulates a measuring system having many sensors (in this case 20 sensors). Each sensor is measuring a temperature in a certain point in some industrial process for example. Temperatures can vary between 0 and 500. You need to write a process control system for this process. The only thing the process control system needs to do is to read temperature values from these sensors. The important thing is that your system needs to work in real-time so that it is capable to read values fast enough (so that it could for example give an immediate alarm if the temperature value goes above a given limit). This means that your application needs to read all temperature values without any long delay.

The multi sensor simulator creates the measurement values. Each sensor sends it values to their own descriptor so that you have 20 sensor descriptors in your program. Your program can read values from the file descriptors that become ready when you have called the function StartSimulator. Each sensor makes it's own decision, when they send a new value to the descriptor. The delay between two values from the same sensor can vary from very short time (about a few milliseconds) to the 9 seconds. And as said before your program needs to receive each value as fast as possible after it is sent from the sensor.

To help to analyze the response times the sensors send a sending time with the value. This means that you can verify how much time has elapsed between sending and receiving the value. Sensors send the data to the file descriptor as a structure that is defined in the following way:

```
typedef struct {
    struct timespec moment;
    int value;
} Tmeas;
```

Immediately after you have read this kind of structure from the sensor file descriptor you can read the current time from the computer and calculate the time difference that indicates the delay of your read. Use the function diff_timespec for that purpose (see MultiSensorSimulator.h). Calculate and display the sum of all delays, because it gives a good overall figure how well your application works in real time. Sum of delays can be calculated with the function increment_timespec (see MultiSensorSimulator.h).

How to use the multi sensor simulator

It is easy to use the multi sensor simulator. Only thing you need to do is to include the file MultiSensorSimulator.h to your source code and link object file MultiSensorSimulatorEdu.o to your object code, if you are working with Edunix computer. If you are working with virtual Linux machine Fedora10, you need to link MultiSensorSimulatorVir.o instead of MultiSensorSimulatorEdu.o.
You can find these files from the web at the web address http://users.metropolia.fi/~hannuvl/ke13/realtime_prog/MultiSensorSimulator/.

If the source file of your application were myapplication.c, then you can compile and link it in the Edunix in the following way:

```
gcc -o myapplication.exe myapplication.c MultiSensorSimulatorEdu.o -lrt
```

In the source code you only need to call function StartSimulator as follows:

```
int noOfSensors;
int sensorDescriptors[20];
int noOfValuesPerDescriptor = X; // what you want

noOfSensors = StartSimulator(sensorDescriptors, noOfValuesPerDescriptor);
// function returns the number of sensors, that is always 20.
// Parameter noOfValuesPerDescriptor specifies how many values you want
// each sensor to send
```

After doing that you can read measurement values from sensors. For example, you could read one measurement value from the first sensor and display it to the screen in the following way:

```
Tmeas measurement;
read(sensorDescriptors[0], &measurement, sizeof(Tmeas));
printf("Measurement value was %d\n", measurement.value);
….
```

When sensor has sent all values, reading the file descriptor of that sensor causes end of file situation. If you ask each sensor to send 5 measurements by calling StartSimulator in this way:

```
noOfSensors = StartSimulator(sensorDescriptors, 5);
```

It takes about 30 seconds from the simulator to send all values. The time can vary, because the delays are random values.

The implementation

This is a teamwork exercise but this topic can also be implemented by one person if desired. The suitable team size for this project is two people if teamwork option is preferred.. In the very beginning the project plan should be done and the whole project should be implemented according to the principles of real project work (See a separate document "Instructions for project work").

The documentation

A project report is composed about the project. A special emphasis in the project report should be put on the following things. You have to explain, what the problem actually is and how you have solved it. It is important to document how the system is constructed as a whole and how it is working as a whole. Why the solution you have used solves the original problem.

One more important thing in the documentation is the description of how the system was tested. It is important to analyze the test results and make conclusions what they really mean. In this particular project you should at least calculate the average delay, maximum delay and minimum delays in your program when receiving measurement values. These results should be documented in the project report and their meaning should be analyzed.

The report should also indicate the responsibilities in the project (who made what in the project) if it is done as a team.

The deadline

The deadline is the 24th of May in 2013.