CSC9006 Real-Time Systems (Spring 2021) Homework 6

** **Submission deadline: 9PM, June 1st, 2021.** **

1. Introduction

In this homework, we will start to modify the TAO real-time event service, and we will learn to use Git to manage our code.

You may get Git for Ubuntu using the following command:

```
$ sudo apt install git
```

Visit its official website to learn more: https://git-scm.com/

In the following, we will cover some Git usage along the way. In addition, here are two useful pages describing

- some basic concept of Git: https://git-scm.com/book/en/v2/Getting-Started-What-is-Git%3F
- how to undo things using Git: https://git-scm.com/book/en/v2/Git-Basics-Undoing-Things

2. Managing our code using Git

We can use Git to take snapshots for each stage of our system development, and we can undo our changes when needed, or even rollback to some earlier snapshot. In this sense, the snapshots of a project make a directed chain graph, where each vertex corresponds to a snapshot.

What's more, using Git we may branch our project to try out some crazy ideas in parallel. In this sense, the snapshots make a directed tree graph, and we may create and switch among branches in the graph, each branch being a directed chain graph of snapshots.

Finally, using Git we may merge our project branches to integrate our development. In this sense, the snapshots make a DAG (directed acyclic graph).

Now, type the following for a quick intro to a set of Git commands:

```
$ man giteveryday
```

Read the section **INDIVIDUAL DEVELOPER (STANDALONE)**, try out those commands to get familiar with them. These are commands useful for this homework assignment, and for everyday life in general.

Now let's create a Git repository for this homework. Switch to directory `$TAO_ROOT/orbsvcs/orbsvcs`, and type the following at the prompt:

```
$ git init
```

`$ git add .`

`$ git commit -m "First commit"`

`$ git status`

If everything works alright, you should see the following output:

```
On branch master
nothing to commit, working tree clean
```

Recall that you can type `man git-'command-of-question'` to learn more about each command. For example, type `man git-commit` to learn more about the `-m` option

Assignment 1 (10 points): Create a new branch named `hw6` and switch to that branch. Write down the Git command(s) you ran to achieve it, and write them into a plain text file and name it `readme.txt`.

If you did it alright, type `git branch` and you should see the following:

```
* hw6
master
```

The asterisk symbol (*) shows which branch we are at right now.

We will work within the `hw6` branch throughout the rest of this homework assignment. In this way, we can make sure that the original TAO (which resides in branch `master`) will be kept intact.

3. Modifying the TAO real-time event channel

In this assignment, we are going to short-circuit the event channel, so that our implementation will bypass all the filtering/scheduling modules. In Homework 7, we will insert a new module to the event channel.

First of all, we will do a small yet important change to the event channel IDL definition. In file `RtecEventComm.idl`, modify the main event delivery callback operation (line 136) by removing label `oneway`, like the following:

```
135    /// Main event delivery callback
136    //oneway void push (in EventSet data);
137    void push (in EventSet data);
```

Then type `make` at the same directory, to trigger TAO's IDL compiler to recompile this definition, and to trigger the g++ compiler to recompile related files. Making this change we can ensure that TAO will never use *oneway invocation* (i.e., best-effort, no guarantee of delivery) to push events to event consumers.

Now it is a good time to use Git to create a code snapshot for this important change.

Type `git add .` at the same directory to stage our change. Then type `git status` and you should see the following:

```
On branch hw6
Changes to be committed:
 (use "git restore --staged ..." to unstage)
        modified:   .shobj/RtecEventCommC.o
        modified:   RtecEventComm.idl
    modified:   RtecEventCommC.cpp
```

> modified: RtecEventCommC.h
> modified: RtecEventCommS.cpp
> modified: RtecEventCommS.h
> modified: RtecEventCommS_T.cpp
> modified: RtecEventCommS_T.h
> modified: libTAO_RTEvent.so.3.0.0
> new file: readme.txt

Then type `git commit -m "dropping the use of oneway invocation"` to commit the change to our repository. You can use `git log` to see all the committed changes. A curious mind should also check out the related IDL definition in `RtecEventChannelAdmin.idl`.

Assignment 2 (20 points): Find out that in file `EC_Default_ProxyConsumer.cpp`, at line 90, for the statement `ace_mon.filter->push (event, this);`, which class' push function will be called? You may figure this out by some code tracing.

Assignment 3 (20 points): Find out which configuration is the default option for the collection of `ProxyPushSupplier` objects? Recall that such a configuration includes three types of flags. You may want to review this page for a list of flags: http://www.dre.vanderbilt.edu/~schmidt/DOC_ROOT/TAO/docs/ec_options.html (Hint: trace the code for the relevant class constructor).

Assignment 4 (20 points): Explain the purpose of the statements in lines 95-99 in file `EC_Default_ProxySupplier.cpp`.

The above assignments should help you get more familiar with the original event channel implementation. Although eventually we will bypass those filtering/scheduling implementation, but it is critical to have some familiarity with them, since they help us to see where and how we should make our changes.

With those preparation, now we are ready to modify the code to short-circuit the event channel. To begin with, add the following code to file `EC_Default_ProxyConsumer.h`:

```
#include "orbsvcs/RtecEventCommC.h"
#include "orbsvcs/ESF/ESF_Worker.h"
```

Also, in the same file, add the following between `TAO_BEGIN_VERSIONED_NAMESPACE_DECL` and `TAO_END_VERSIONED_NAMESPACE_DECL`:

```
class TAO_EC_HW6_Worker : public TAO_ESF_Worker<TAO_EC_ProxyPushSupplier>
{
public:
  TAO_EC_HW6_Worker (const RtecEventComm::EventSet event);
  virtual void work (TAO_EC_ProxyPushSupplier *supplier);
private:
  RtecEventComm::EventSet event_;
};
```

Then, in file `EC_Default_ProxyConsumer.cpp`, add the following between `TAO_BEGIN_VERSIONED_NAMESPACE_DECL` and `TAO_END_VERSIONED_NAMESPACE_DECL`:

```
TAO_EC_HW6_Worker::TAO_EC_HW6_Worker (const RtecEventComm::EventSet event)
  :  event_ (event)
{
}
void TAO_EC_HW6_Worker::work (TAO_EC_ProxyPushSupplier *supplier)
{
  supplier->reactive_push_to_consumer (supplier->consumer(), this->event_);
}
```

And in the same file add `#include "orbsvcs/Event/EC_Default_ProxySupplier.h"` as well.

Assignment 5 (30 points): In file `EC_Default_ProxyConsumer.cpp`, remove everything within function `TAO_EC_Default_ProxyPushConsumer::push (const RtecEventComm::EventSet& event){...}`, and add your code within this function to short-circuit the event channel. That is, your code should push every event to every consumer that has connected to the event channel. You should type `make` at the parent directory to build your code into an updated library, and then you may use the example at directory **$TAO_ROOT/orbsvcs/examples/RtEC/Simple** to verify your implementation. A correct implementation should give the same result as the original event channel did, and there is no need to modify the example.

Here is some guidance for you:

- Remember that in the event channel, `ProxySupplier` handles consumer clients, while `ProxyConsumer` handles supplier clients.
- Consult the on-line documentation often: https://www.dre.vanderbilt.edu/Doxygen/7.0.0/html/; in particular, consult both entry `TAO_RTEvent` and entry `TAO_ESF`.
- Read this to see what a Worker class is : https://www.dre.vanderbilt.edu/Doxygen/7.0.0/html/libtao-doc/esf/a05661.html#details .
- Read the comment in file `EC_Supplier_Filter.h`. Alternatively, you can also read the corresponding page in the on-line document (since the on-line document is automatically generated from those comments).
- Recall that by default, the event channel can be configured to make some of its components either null or trivial. You may learn from those implementations.
- Besides using Cscope, you may want to add some `printf()`s here and there to help you visualize where the program jumps. Also, the Linux utility `grep` is a great tool to search for keywords.
- Finally, make some good use of Git so that you do not mess up the code while investigating/modifying the code.

4. Moodle submission

Write you answer for Assignments 1-4 into the `readme.txt` file. Clearly label each of your answer. Package the following two files into one single zip file, and name the zip file using your student ID:

- (70 points) Your `readme.txt`;
- (30 points) Your `EC_Default_ProxyConsumer.cpp`.

Post your questions on the Moodle forum. **Submission deadline: 9PM, June 1st, 2021.**