CSIM Helpful Hints for PS3

This document is simply intended to save you some time by pointing out the features of CSIM which are most relevant to PS3. However, for definitive explanations of any CSIM functions, or data structures always look to the users manual:

`cdf: /local/doc/csim/rev16-docs/users.doc`

The first thing to consider when you start is how many different process types you want. Try to consider the main actors in the simulation and write a process describing the behaviour of each. If you give some thought to your processes at the start

Think carefully about how you would like to represent the different objects and conditions in your simulations. Some of them should be represented as `FACILITY` objects and others as `EVENT` objects.

Sample code for much of the csim functionality can be found in the directory:

`cdf:  /local/doc/csim/examples/c/`

This is a good place to look to learn the proper use of data csim routines and data structures.

Here are some of the particular features of csim that you will find useful in your assignment:

Random Number Streams:

There are several events in the problem which have an associated probability distribution. It is important that the random numbers generated for each of these things is independent. For that reason you should use the random number stream facilities of csim to generate a separate stream of numbers for each distribution. While the streams facility is not documented a quick glance at the example file `do_strea.c` will show you how they work.

Rerun():

The `rerun()` function allows you to clean up after a run and start over from scratch. It takes care of things like deallocating processes, facilities, and tables which were allocated *after* the `create("sim")` statement. Note that tables which are created before that point continue to exist, but their values are cleared. Only `TABLES` (and `QTABLES`) created before the first create statement using the `permanent_table("TableName")` function will persist with their values intact.

I suggest that you structure your program as a loop over the number of simulation runs. Put the `create("sim")` statement at the start of the loop and `rerun()` at the end, with the body of your simulation in between. Allocate all your data collection objects outside the loop. That way they will be available to you at all times. Use two sets of tables:

one non-permanent set for computing statistics within a run, and another permanent set for compiling statistics across runs.

Data Structures (FACILITY, STORE and EVENT):

These are the three objects which you need to make use of to allow the processes to signal information about the state of various objects to each other.

`EVENTS` are occurrences at a particular point in time. An event is signalled by the function call `set(ev)`. Processes can wait on an event which occurs at a particular instance of time. All processes which call `wait(ev)` will block until the event is triggered by some process calling `set(ev)`. At that point all processes which are waiting on the event resume their processing. A useful variant of the `wait(ev)` command is the `timed_wait(time,ev)` command. This function will only wait a specified length of time. `timed-wait` returns a flag indicating whether it stopped because the event occurred or because time ran out.

A `FACILITY` is better for representing resources which only one process may be using at a time. When a process calls the function `reserve(facil)` and other processes have already reserved that facility, then it blocks until `release(facil)` has been called by those other processes which were in front of it. This is why `reserve(facil)` is different than `wait(ev)`, and it is a good way to coordinate queues.

A `FACILITY` can also be used for representing an ongoing condition by using the `use(facil, time)` function which will cause the facility to have a state of `BUSY` during that time. Other processes can periodically test the state of the facility, of the facility without blocking, by using the check `status(facil) == BUSY`. The difference between `use(facil, time)` and `reserve(facil)` is that the former causes simulated time to pass while the latter does not.

`STORE` is another data structure which you might find useful. This is like a facility, except it represents a resource which has slots for multiple entities at a time, whereas a `FACILITY` can only accept one entity at a time. A `STORE` is initialized with the `store("name", size)` function, where size specifies how many spaces there are in the STORE. A process can request a place in the `STORE` by using the `allocate(st)` command. If there is an available space then the count of objects in the `STORE` will be incremented and the process can continue it's execution. Otherwise, the process will be blocked until a space becomes available. Also, the availability of space can be checked without blocking by calling the `avail(st)` function. A process can free up a place in the `STORE` through the `deallocate(st)` command, which then decrements the count of objects in the store.

Other Operations:

Another very important function is `hold(time)` which also causes simulated time to pass.

Also, the function `simtime()` will return a `float` value with the current simulation time. Taking the difference between two simtime functions will give the length of simulation time a particular operation took.

Statistics (TABLE, QTABLE):

Most of your data collection needs can be met by a plain `TABLE`, but a `QTABLE` is better for storing information about queues. While you need a `FACILITY` to implement the functionality of a queue you can use the `QTABLE` data structure to easily record statistics about your queue. Simply call `note_entry(qtable)` when an entity enters the queue and `note_exit(qtable)` when it exits the queue. Then you can ask the `QTABLE` about the mean queue length by calling `qtable_qlen(qt)`.

Values are entered into tables using the `record(val,table)` function, and later you can compute the statistics that you need using `table_mean(table)`, `table_var(table)` etc.

I hope this information is helpful. Good Luck.