libcppsim: A SIMULA-like, Portable Process-Oriented Simulation Library in C++

Moreno Marzolla

Dipartimento di Informatica,
Università Ca' Foscari di Venezia
marzolla@dsi.unive.it
http://www.dsi.unive.it/~marzolla

Motivations

- Yet Another Simulation Library?
- Many of the existing tools and libraries suffer some limitations
  - Special-purpose
  - Complex to use
  - Steep learning curve (new languages to learn)
  - Not very efficient

General principles / 1

- Process-orientation vs Event-orientation
  - Process-orientation: the simulation model is described as a set of (pseudo)parallel cooperating simulation processes
  - Event-orientation: the simulation model is described as a set of events which may alter the state of the simulation
- Process-orientation makes writing complex simulation models easier...
- ...but writing good PO simulation engines is more difficult than EO ones

General principles / 2

- The C++ language was selected for different reasons
  - Simulation models exhibit an inherently object-oriented structure (Simula was the first OO language)
  - C/C++ compilers are quite efficient and produce good code
  - The C++ language is very popular among software developers

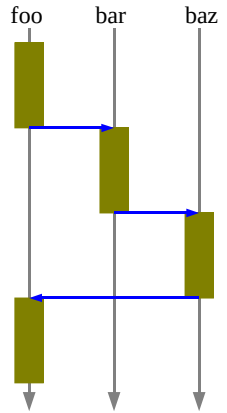
Implementation issues

- Simulation processes are usually implemented in C++ as a collection of OS processes or threads
  - Context switching is not done very efficiently on some OSes
  - C++ does not provide any support for concurrency
  - Simula supported *coroutines*: blocks of code which can be suspended and reactivated later

libcppsim

Coroutines

```
coroutine foo;
begin
   ...
   resume bar;
   ...
end

coroutine bar;
begin
   ...
   resume baz;
   ...
end

coroutine baz;
begin
   ...
   resume foo;
   ...
end
```



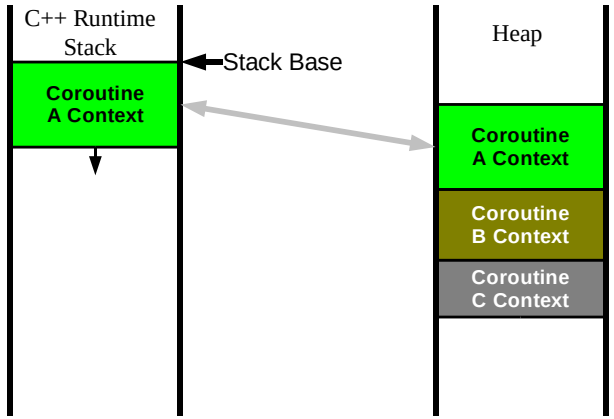
libcppsim

Coroutines in C++

- We implemented the coroutine class to provide the main abstraction of libcppsim
- Coroutines are implemented using the makecontext/setcontext or setjmp/longjmp Unix system calls for user-level threading

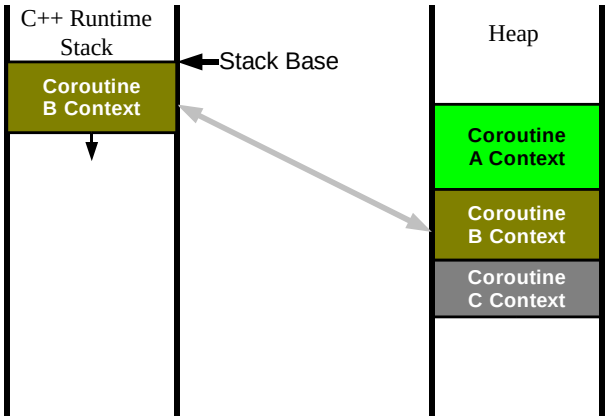
libcppsim

setjmp/longjmp



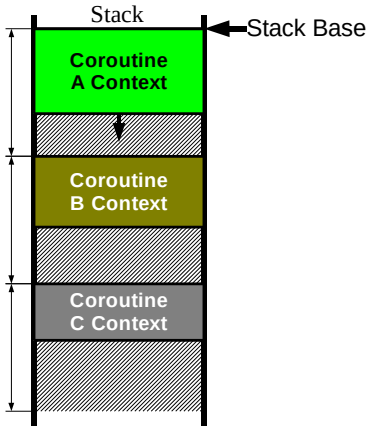
libcppsim

setjmp/longjmp



libcppsim

makecontext/setcontext



libcppsim

Example

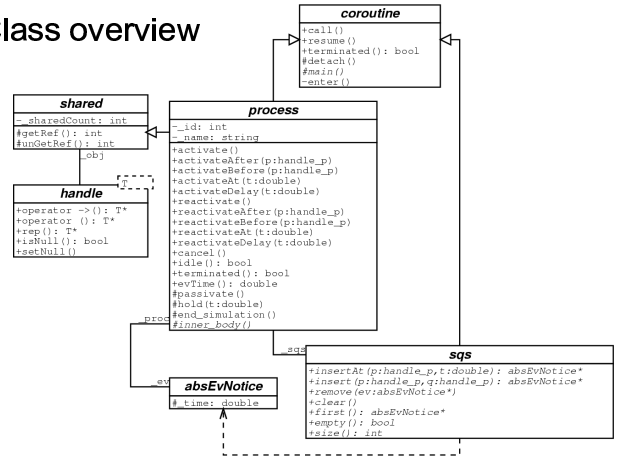
```
coroutine *A, *B;

class cor_A : public coroutine {
public:
    cor_A( void ): coroutine( ) {};
    virtual ~cor_A( ) { };
protected:
    void main( void ) {
        while( 1 ) {
            B->resume( );
        }
    }
};

class cor_B : public coroutine {
public:
    cor_B( void ): coroutine( ) {};
    virtual ~cor_B( ) { };
protected:
    void main( void ) {
        while( 1 ) {
            A->resume( );
        }
    }
};
```

```
int main( void )
{
    A = new cor_A();
    B = new cor_B();

    A->resume( );

    return 0;
}
```

libcppsim

libcppsim Architecture / 1

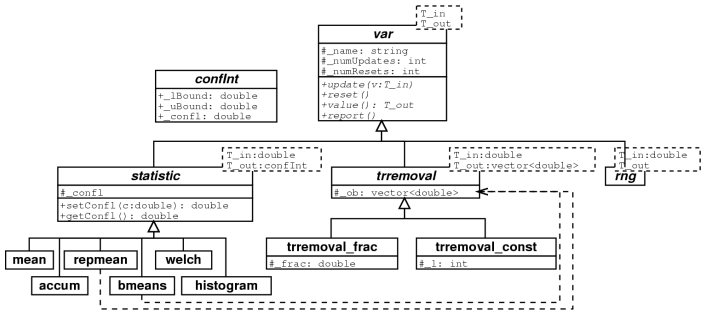
- Class overview



libcppsim

libcppsim Architecture / 2

- Statistics



Using libcppsim

```
class job : public process { ... };

class source : public process {
public:
    source( const string& name ) :
        process( name ) { };
    virtual ~source( ) { };
protected:
    void inner_body( void ) {
        handle<job> j;
        for ( int i=0; i<10; i++ ) {
            hold( 5 );
            j = new job( "job" );
            j->activateAfter( current() );
        }
    };
};
```


A real application

- libcppsim has been used to build UML- , the UML Performance SImulator
  - A software tool for deriving a simulation model from annotated UML specifications
  - UML elements are mapped into simulation processes
    - Deployment diagrams — Resource processes
    - Use Case diagrams — Workload processes
    - Activity diagrams — Action processes
  - Computed parameters
    - Mean execution time of actions
    - Utilization and throughput of resources

Conclusions and future work

- We implemented libcppsim, a process-oriented simulation library in C++
  - Portable
  - Lightweight
- Todo
  - Build higher-level simulation primitives on top of those provided by the library (similar to the DEMOS package for Simula?)
  - Provide a better interface for collecting and reporting statistics
  - Provide a user-friendly debugging support, other than the basic tracing facility already implemented

Availability

- The source code for libcppsim and UML- is abailable on the WEB at:

  http://www.dsi.unive.it/~marzolla

Libcppsim