COROUTINE
User Guide

Version 1.0 (May 1999)

by
Keld Helsgaun
E-mail: keld@ruc.dk

1. Introduction

Coroutines provide the means to organize a program execution as several sequential processes.

A *coroutine* is an object that has its own stack of procedure activations. A coroutine may temporarily suspend its execution and another coroutine may be executed. A suspended coroutine may later be resumed at the point where it was suspended.

COROUTINE is a library for coroutine sequencing in the programming language C++. The facilities of the library are based on the coroutine primitives provided by the programming language SIMULA.

The coroutine library has been used to implement a C++ library for process-oriented discrete event simulation. This library is also provided along with libraries for random drawing and handling of two-way lists.

Also provided is a library for the combination of coroutine sequencing and non-determinism.

The library is described in the report

> K. Helsgaun,
> "A portable C++ library for coroutine sequencing",
> Roskilde University, 1999.

This report may be found in pdf-format in the directory DOC.

2. Installation

The software is available in two formats:

COROUTINE-1.0.tgz	(gzipped tar file, ≈ 180 KB)
COROUTINE-1.0.sit	(Stuffit archive, ≈ 180 KB)

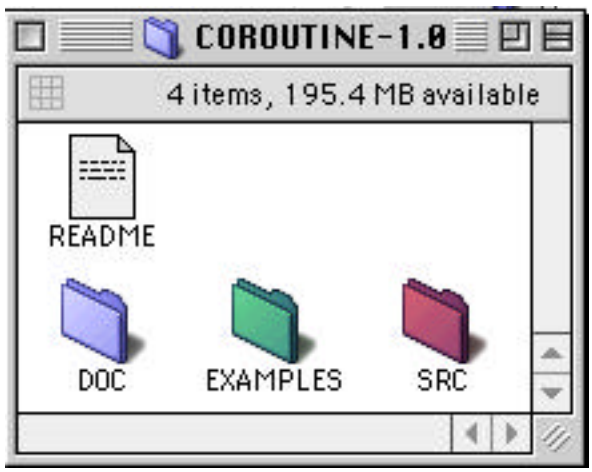
If a UNIX machine is used, download the software in the first format. Next execute the following UNIX commands:

```
gzip -d COROUTINE-1.0.tgz
tar xvf COROUTINE-1.0.tar
```

If a MacOS or a Windows machine is used, download the software in the second format. Next unstuff it with StuffIt Expander™ (freeware available at http://www.aladdinsys.com).

The code is distributed for research use. The author reserves all rights to the code.

On a MacOs machine the following files and folders can be found in the folder COROUTINE-1.0. Similar files and directories can be found on a Windows machine and on a UNIX machine.

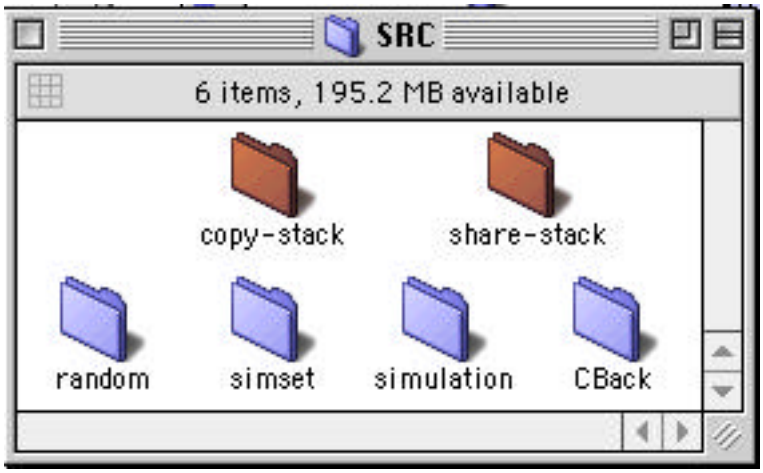


The `README` file contains instructions for installing the software.

The `DOC` folder contains the following documentation: (1) `COROUTINE_GUIDE.pdf`, this user guide, and (2) `REPORT.pdf`, a report that describes the use and implementation of the software.

The `EXAMPLES` folder contains all the example programs from the report.

The SRC folder contains source code. On a MacOs machine the folder has the following appearance.



The coroutine library is provided in two versions. The versions are made available in two separate folders: `copy-stack` and `share-stack`. Each folder contains a header file, `coroutine.h`, and a source file, `coroutine.cpp`.

The remaining four folders, `random`, `simset`, `simulation` and `CBack`, contain header and source files of the random drawing, list handing, simulation and backtrack programming library.

Both versions of the coroutine library should run without modifications on most platforms. However, for the copy-stack version a small adjustment may be necessary. Some C++ systems do not always keep the runtime stack up-to-date but keep some of the variable values in registers. This is for example the case for C++ systems on Sun machines. If this is the case, the macro `Synchronize` must be used. The comment characters are simply removed from the macro definition (in the beginning of `coroutine.cpp` of the copy-stack version).

The library may now be compiled and tested with the program `TestProgram.cpp` in the `EXAMPLES` folder.

The program should produce the following output

```
m1a1m2b1m3a2c1a3b2c2
==>
```

and wait for input from the keyboard.

If the character `r` is typed, the program should print

```
b3a4m4c3m5
```

and stop.

If the character `c` is typed, the program should print

```
b3a4c3m4c4m5
```

and stop.

If any other character is typed, the program should print

```
m4c3m5
```

and stop.