

Lund Simula for Macintosh/MPW ver 4.07

Users manual notes

The Users manual is distributed as part of the Hypercard reference stack. This is an extract for those who have not Hypercard available.

Scripts for compilation and linking

- **simcomp** [-p -l -e -s -n -w -h -z -x] <file>

simcomp invokes the compiler on the file <file>.sim and the assembler on the result to produce the file: <file>.sim.o. Options:

- p (Default) Print short message after each compiler pass.
- l produce a profram Listing with merged error messages on file <file>.lis
- e redirect the error listing to the file <file>.err
- s (Default) produce symbolic assembly code
- n (Default) do not produce binary code
- w inhibit Warning messages
- h inhibit generation of linenumber Hooks (faster execution but less informative error messages).
- x generate error messages when error is discovered. Errors are normally sorted and shown to the user during pass 4.

- **simld** <main> <module1> <module2> ...

simld invokes the linker to combine the SIMULA main program <main>.sim.o with the separately compiled classes/procedures in <module>.sim etc. (if any) into the executable program <main>.

- **sim** <main> <module1> <module2> ...

sim works as a combination of **simcomp** on each of the modules (in backwards order) and then a **simld** on all of the files. **sim** is specially handy to compile and link single module programs.

- <main> [-p -g -k=kn -m=mn -input=<file> -output=<file>]

A simula program is executed by simply entering its name. Options:

- p Print a message and some statistics when execution starts and finishes.
- g Write a message each time the garbage collector is called. Useful when tuning memory pool.
- k=kn Execute the program with kn KBytes of pool memory. Default is 512KByte.
- m=mn Execute the program with mn Megabyte of pool memory. Note: giving the program too much memory might result in system hangs since no space for system code might be left over.
- input=<file> read sysin input from <file>. This is an alternative to shell level I/O re-direction.
- output=<file> write sysout output to <file>.

Segments and limitations

The SIMULA system conforms to the SIMULA standard and there are no restrictions in the compiler. Unfortunately the segmentation of the program forces some limitations on the size of Simula programs. It is our hope that in later versions of the SIMULA system we will be able to relax these restrictions.

An executable file is divided into one data segment and one or more code segments. The data segment is restricted to 32 Kbytes. All data is automatically placed in the data segment by the system. The code segments can be bigger than 32 Kbytes but there are restrictions (see below). All communication between different segments is done via a jump-table. The number of entries in this table is maximized to 8192. In Simula calls between separately compiled units and calls to the run-time system will use entries in this table.

The assembler is responsible for the partition into code segments. When it starts to assemble a new Simula block (procedure or class) it will generate a new segment if the old one is bigger than 16K.

- The generated code for a block (sub-block, procedure or class) may not be bigger than 32 Kbytes. This will not be a very hard restriction. In a reasonably formatted program it means that a block should not be more than around 1000 source lines. The linker will issue a message if this restriction is violated.
- The 32 Kbytes data segment gives some restrictions. The SIMULA system uses the data-segment for three different purposes:
 - for local variables of the run-time system
 - for templates (a description of a block used by the run-time system)
 - for strings and floating-point numbers used in the Simula program

This is probably the most severe restriction. Depending on the program, it means that a Simula program must not be bigger than between 4000 and 10000 lines of Simula source.

- The jump-table is used for a number of internal labels in the run time system and for all entry points in the Simula program (every procedure and block has one entry point; a class has at most three entry points). This restriction has not yet been a problem.
- Labels which have to be exported from its segment can not have a relative address in their segment greater than 32K. The assembler tries to divide the program into segments not greater than 32 Kbytes. However if a block is greater than 16 Kbytes the whole segment can be greater than 32Kbytes. There are two different cases where this situation can cause problems.
 - If a label is used in a block different from the block with the declaration and it has a relative address greater than 32 Kbytes.
 - If an entrypoint after an "inner" statement has a relative address greater than 32 Kbytes.

Other restrictions

- the symbolic debugger is not a part of this version of the system.
- a file name may not contain blanks

Filetypes

Files created by a Simula-program are **not** of type TEXT. In order to treat such a file as text the file type must be changed with the command:

```
setfile -t TEXT filename
```

Windows and Worksheet

There are a few things that make MPW different from other systems. Every window has an editor but can also be used to execute MPW commands (that is what we do when we execute the commands in the comment in the program SimulaHasClass.sim). The Simula sysin and sysout files are always connected to the window from which the execution is started. The editor in the window has a bar-sign (|) at a point where a new character should be inserted. Also sysout uses this point.

There are two different ways to direct input from the window to the Simula sysin image.

- If there is no text which is marked (written in inverse video) then the whole line containing the bar is read. The reading starts when ENTER is pressed. (If CR is pressed the bar is moved to the next line but nothing is read.)
- If a section of text is marked then this piece of text is read as soon as ENTER is pressed. The text is read exactly as it is written with each line filling subsequent images.

When using breakoutimage to display a prompt the user must mark the entered text before hitting ENTER, otherwise all text on the line (including the prompt) will be read by the Simula program.