903 ALGOL SEPARATE COMPILATION.

Overview.

Translating in library mode causes an ALGOL program to be translated in a way that expects a library tape to be offered when the halt is reached at the end of the source tape. Continuing from this halt by an entry at 9 performs a library scan that just copies the needed procedures from the library (relocatable binary) tape into the translated program. Reading is at full speed from the start of a procedure until the blank tape at the end of each procedure. (This is illustrated in HUNTER ALGOL DEMO7 and DEMO8).

If no library material is needed then the halt for a library scan is not reached and the effect is that more space for code and data is available.

Elliott ALGOL comes with a default library tape that contains the standard functions and procedures described in the section on language features and facilities. Users can extend this library tape by inserting relocatable binary code compiled using a 1-pass SIR assembler. This is described in the section later on writing code procedures for ALGOL.

Pre-compiled ALGOL procedures

ALGOL programs can also be translated to paper tape, and the output paper tape altered into a suitable relocatable binary (RLB) intermediate tape for linking to the calling program when using the 2-Pass and Load-and-Go systems. The Large Program systems can only load ALGOL as a main program and not as a code procedure for technical reasons to do with how the Large Program systems allow compiled ALGOL code to straddle store module 0 and 1.

Loading of pre-compiled ALGOL procedures is facilitated by an ALGOL program in a script called MODULE.DAT (in the directory HUNTER) that reads in a paper tape from its initial blank section up to the first of the six trailing erase signs at the end. It then thinks for a while and outputs the RLB material.

The script TAN.DAT (in the directory HUNTER) shows how a precompiled procedure can be transformed into a pre-compiled "module". Notice that TAN.DAT has a "JUMP 17" to cause translation to paper tape, and that the paper tape output is directed to a (binary format) file TAN.BIN. Notice also the semicolon at the end of the procedure. A semicolon is essential here, and also at the end of a procedure with begin and end which is the normal case.

The output from translating the procedure contained in TAN.DAT is in TAN.BIN. Notice the 50 or so rows of blank tape (zeros) at the start, the loader halt code sequence(112 0 0) near the end, the 50 blanks following the loader halt code, the 6 erase characters (255) and finally the 100 rows of blank tape at the very end. When you write a separately compiled procedure, check that its output ends with a set of 255 numbers. If not there is definitely an error so check with a normal compilation (entry at location 8) until you have cured the error.

The TAN.DAT script then runs the MODULE program on TAN.BIN to create data suitable for adding to a library. Note that the MODULE.DAT script reads in a binary format file as text, and likewise emits a binary as text, overriding the default handling of .BIN and .RLB file suffixes. This is a convenient way to simplify the handling of binary files.

(Note it is important that MODULE.DAT sees a 255 number at the end because it reads in data blindly until it finds a 255 before processing any data. It fails by trying to read beyond the end of the tape when there is no 255 number present).

Pre-compiled procedures can be added to programs by one of two methods: they can be loaded as relocatable binary tapes at runtime (e.g., by entry at 14 in the load-and-go systems) or they can be edited into a library tape suitable for scanning in library mode.

An example use of entry at 14 to load precompiled procedures in addition to the standard library is given in DEMO8.DAT in the directory HUNTER.

However, with trivial editing using a program like NOTEPAD on binary format files, a file like TAN.RLB can be inserted into a library which is more convenient if it is to be used in several programs. The main requirement is to put the inserted file content just ahead of the trailing sequence of characters

112, 0, 0. If desired, additional 0's can be added to the front and back of the RLB to align layout. This has already been done for the file LIBRARY.BIN in the HUNTER directory.

There are three examples of the use of pre-compiled procedures and library mode, in the files EXAM1.DAT, EXAM2.DAT and EXAM3.DAT in the HUNTER directory. Each can be run as it stands.

EXAM1 uses the tan(x) procedure described above. There is an odd bit of EXAM1 where it declares a code procedure, qatrig, and then mentions qatrig in a statement that can never execute it. The purpose of this is that sin and cos are located together in qatrig, and that without this trick they would not get loaded because they are not explicitly mentioned in EXAM1, even though they are called in tan.

EXAM2 replaces simps in HUNTERR\DEMO4 by a separately compiled one in SIMPS.BIN prepared from SIMPS.DAT. Now you see the limitations appear - you can't have constants! Actually you ARE allowed constants 0, 1 and 3, and must build up things like 2 and 4 by the trick inside SIMPS.DAT.

In fact you can have ludicrous constants, but only if the calling program and the separately compiled procedure mention them in a strictly identical order, e.g., in main you could have

```
dum := 3.14159 + 2.71828;
```

and in the separate bit you can have

```
fred := 3.14159 * joe;
```

but that's silly isn't it ?

Neither can you have switch lists nor labels but that is no hardship. If you do then MODULE.DAT reports an error.

EXAM3 is a real program for finding out whether a billing program began to run faster by fitting some normal equations to the data using code written by T.J. Dekker at the Amsterdam Mathematisch Centrum in 1963 (contained in DET.DAT and SOL.DAT), and a couple of toy procedures in COPY.DAT and SIGMA.DAT. Here is another giant snag though - the called procedure, DET, uses space in QAVNDA, in fact 16 words, so the calling program goes and declares 8 dummy reals! This is still simple compared to what might be necessary in a chain of calls.

How does one know that det uses 16 words? Well there are two needed for each real, one for each integer and 2*d+2 for each d-dimensional array.

EXAM3 pretends to call sqrt because sqrt is called within det, and would not be loaded otherwise.

Building procedures into system tapes.

Precompiled procedures can be added to the two pass system as follows.

To build a procedure or function into TAPE 1, enter the translator at 10 with a program in the form:

```
title;
begin code real procedure random(i); integer i; algol;
```

as input and then enter at 16. A new TAPE 1 embodying the routine will be output. (The corresponding intermediate code for the routine must be added to the library tape for the interpreter.)

A dump of TAPE 1 can be obtained by entry at 8001.

To build a routine into TAPE 2, load a library tape with a copy of the routine included at 12, then dump at 14.

To add a routine to the library tape (TAPE 3), copy the tape excluding the terminating character 01110000 (decimal +112), append the routine relocatable binary, then the terminating character.