SLAC-PEP-NOTE--369

DE89 006358

FEB 0 1989

GO, AN EXEC FOR RUNNING THE PROGRAMS: CELL, COLLIDER, MAGIC, PATRICIA, PETROS, TRANSPORT, AND TURTLE*

Hamid Shoaee
Stanford Linear Accelerator Center
Stanford University, Stanford, California 94305

INTRODUCTION

An exec has been written and placed on the PEP group's public disk (PUBRL 192) to facilitate the use of several PEP related computer programs available on VM. The exec's program list currently includes: CELL, COLLIDER, MAGIC, PATRICIA, PETROS, TRANSPORT, and TURTLE†. In addition, provisions have been made to allow addition of new programs to this list as they become available (see Appendix B).

The GO exec is directly callable from inside the Wylbur editor (in fact, currently this is the only way to use the GO exec.) It provides the option of running any of the above programs in either interactive or batch mode. In the batch mode, the GO exec sends the data in the Wylbur active file along with the information required to run the job to the batch monitor (BMON, a virtual machine that schedules and controls execution of batch jobs). This enables the user to proceed with other VM activities at his/her terminal while the job executes, thus making it of particular interest to the users with jobs requiring much CPU time to execute and/or those wishing to run multiple jobs independently. In the interactive mode, useful for small jobs requiring less CPU time, the job is executed by the user's own Virtual Machine using the data in the active file as input. At the termination of an interactive job, the GO exec facilitates examination of the output by placing it in the Wylbur active file.

*Work supported by the Department of Energy, Contract DE-AC03-76SF00515.

†The programs CELL and COLLIDER are not available on PUBRL 192; for information on using these programs please contact Helmut Wiedemann.

Notational Conventions

The editorial conventions used in this document are as follows:

1 Go exec prompts and computer messages appear in upper case letters.

2 User inputs are shown in lower case letters.

3 The lines in brackets ([...]) are annotations and are not part of the console display.

4 The upper case letters of a command represent the minimal abbreviation, e.g., TRANsfer.

5 The symbol <CR> corresponds to pressing the carriage return (or the ENTER) key.

Requirements for Using the GO Exec

In order to use the GO exec and the related computer programs, the user should have access to the PEP public disk. To do this on a permanent basis, insert the following line in your PROFILE EXEC A file:

```
exec gime pubrl 192 p (rep
```

In addition, it is assumed that the user:

- is familiar with VM file naming convention and file management commands such as Listfile, ERASE, COPY, AND RENAME.
- can use the VM Wylbur editor.
- has some unused space on his/her A disk.
- has either another permanent disk or a temporary one for saving the output of programs running interactively.

INTERACTIVE EXECUTION

The simplest way to use the GO exec is to collect the input data file for the desired program in the Wylbur active file --by either creating a new data file with the Wylbur Collect command, or USEing/MODifying an existing one-- and then issue the command GO. You will then be prompted for the program and execution modes and options. Following is a list of all of the GO exec prompts with an explanation of the options list. (In what follows, the GO exec prompts are capitalized to distinquish them from the explanations; also, the symbol <CR> corresponds to pressing the carriage return key in response to some of the prompts.)

Upon invoking the GO exec you will be served a menu of the available programs and will be requested to make a selection:

```
PROGRAM MENU:

          1 . . . CELL
          2 . . . COLLIDER
          3 . . . MAGIC
          4 . . . PATRICIA
          5 . . . PETROS
          6 . . . TRANSPORT
          7 . . . TURTLE
          8 . . . DUMMY1
          9 . . . DUMMY2

INPUT THE NUMBER FOR THE DESIRED PROGRAM, OR <CR> TO EXIT:
```

You could at this point either type the number corresponding to the desired program (and for which you already have the input in your Wylbur active file) or press carriage return (<CR>) to exit the GO exec and end up back in Wylbur and with your active file intact. Assuming you picked a program to run, you will next be prompted for the execution mode:

```
             IS THIS A BATCH OR AN INTERACTIVE JOB?
             B = BATCH, <CR> = INTERACTIVE.
```

To run your program interactively, i.e. on your own Virtual Machine, just press <CR>; you will next be asked to specify the disk on which the output file will be saved:

```
             SPECIFY DISK TO SAVE OUTPUT FILE ON (<CR>=B)
```

You should type the one letter mode of the desired disk (e.g. A, B, ...etc.) A carriage return (<CR>) will designate your B disk (if you have one) as the output disk. If you anticipate a large output file, it is best not to save it on your A disk as this may easily fill the disk and cause an abnormal termination of your program. You could alternatively obtain a temporary disk before calling the GO exec and use that for output storage (see Appendix A).

At this point your active file is saved on the above disk as:

```
                    Program_Name INPUT disk_mode
```

where Program_Name is the name of the program you have selected to run, and disk_mode is the mode letter of the output disk. For example, if you are running a PETROS program, and are saving the output on your C disk, then your active is saved as:

```
PETROS INPUT C
```

The one execption to the above naming convention is TRANSPORT: since under CMS file-names are limited to a maximum of 8 characters, the input file for TRANSPORT is saved as

```
TRANS INPUT C
```

The GO exec subsequently LOADS and STARTS the program using the above data file as input. At the completion of program execution --which might be awhile, depending on the program and the system load factor-- the Wylbur editor is called and the output file which has been saved on disk is placed in the active file. The naming convention for the output file is identical to that of the input files described above; however, as some programs generate more than one output file, the filetype has been chosen to reflect the logical unit number to which the file has been written. For example, the program PETROS contains the following output statements:

```
WRITE(6, )
WRITE(20, )
```

Hence, the output files are named: (assuming output disk is C)

```
PETROS OUT06 C
PETROS OUT20 C
```

To get a printed copy of one of the above files, type:

```
print petros out06 c (cc
```

The letters cc indicate that the file contains carriage control characters and MUST be included in the print statement, otherwise an error message will be issued. It should also be realized that a subsequent interactive execution of any program will cause any existing output files previously generated by that program to be overwritten and thus permanently lost. To keep an output file from being destroyed, you could either RENAME, PRINT, or COPY it to another disk. In addition, if you have specified a temporary disk for output, the entire disk and thus any file saved on it will be lost when you LOGoff (see Appendix A).

Interrupting Execution of an Interactive Job

To abort execution of an interactive program or the GO exec, hit the BREAK key ONCE ONLY. The computer will respond with either a "?" or a "!"; type "hx" and press return. The job is now canceled and you are back in CMS. The partially written output files, if any have been generated, could now be examined with the WYLbur USE command.

If you hit break too many times in rapid succession, you might be put in CP enviroment; type "Begin" to return to CMS.

BATCH EXECUTION

As mentioned earlier, the batch option allows for both unattended execution of long runs and submission of multiple jobs. The steps involved in running a batch job are similar to those for interactive execution: collect the input data file into the Wylbur active file and then type GO. In response to the menu prompt you could again either type the number corresponding to the program to be executed, or type <CR> to exit the GO exec and end up back in Wylbur. Assuming you have picked a program to run, the rest of the promts will be as follows:

```
IS THIS A BATCH OR AN INTERACTIVE JOB?
B= BATCH, <CR>= INTERACTIVE.
```

Type "b" for batch job.

```
SPECIFY DISK TO SAVE OUTPUT FILE ON (<CR>=B)
```

Input the one letter mode of the disk on which your active file will be saved using the naming convention mentioned above; B is the default disk.

```
RUN HOLD? YES, OR <CR> FOR RUN
```

If you respond with either "yes" or "y", the output of the batch job will end up in your reader and can then be reviewed by using Wylbur's FETch command (see below). A <CR>, "no", or "n" response will send the job output to the printer and, ultimately, to your assigned bin.

```
SPECIFY JOB CLASS: (TIME LIMITS ARE IN CPU MINUTES)
X=1; S=2; B=4; M=8; L=30; J=120; (<CR>=X).
```

Job class indicates the maximum amount of CPU time a job should be allotted; the response should be either <CR> for class x, or for longer runs, any of the classes s through j. Most of the programs on the menu typically run in less than 1 minute, but can require more time depending on the input data. Class x jobs have the highest priority and are usually executed immediately.

Once the job has been submitted (it could take a few seconds) you recieve a message like:

```
PUN File 4687 to BMON COPY 001 NOHOLD
JOB HXS139 CLASS X, Time 60 Seconds Submitted
```

indicating that your job has been submitted. Each batch job has a unique jobname formed of your account-id (not VM user-id) and a three digit number (HXS139 in the above example.) The number starts from 001 and is incremented each time you submit a job. The jobname is used to query the status of or cancel your job. When BMON (the batch monitor) logs your job in the job queue, it sends you the message:

```
MSG FROM BMON: JOB job_name ACCEPTED
```

Similerly, at the termination of your job, you receive the following messages from a batch machine

```
FROM BATCHnn: JOB 'jobname' ENDING... IN BATCHnn, CLASS X
FROM BATCHnn: BATCH RETURN CODE= 0
                    .
                    .
                    .
```

Other lines indicate the amount of virtual and total time your job used, the number of records read, punched, and printed, etc. Finally, you receive more messages informing you of any job output file being sent to your reader (if you specified RUN HOLD) and a CON file which contains the console displays that occured while the job ran. A RETURN CODE of 0 indicates successful completion of your job. For a nonzero RETURN CODE you should review the CON file to determine the cause of your job's failure.

Inquiring About Batch Jobs

The command for querying the status of your batch job is:

```
JOBSTAT qualifier class
```

Only the first character of the qualifer must be typed. The qualifiers are:

L	Locate jobs
Q class	query the job Queue (class optional)
A class	query Active jobs (class optional)

Class refers to one of the BATCH job classes X,S,B,M,L,J (see above), and where it can be specifed, the default is ALL.

Examples:

```
jobstat l       [locate user's not-completed jobs]

jobstat a b     [show jobs active in class b]

jobstat q       [show queued jobs for all colsses]

jobstat q x     [show queued jobs in class x]
```

Cancelling a Submitted Batch Job

A submitted batch job could be terminated at ay time before its completion with the BATCH CANCEL command. The format of the command is:

```
BATCH CANCEL jobname
```

Where "jobname" is the unique indentifier of a batch job as described above. For example, to cancel the job "HX5123" issue:

```
batch cancel hx5123
```

Use jobstat to locate all the jobs for your user-id (see above). The BATCH CANCEL command cancels a job whether it is waiting to execute or is executing when the command is received. If it is executing, the job is terminated abnormally and the partially written output files are "DISK DUMPed" to your reader; to examine these files, see the following section on retrieving batch output.

Retrieving Output of Batch Jobs

If you requested your job to be processed RUN (nohold), the printed output will ultimately end up in your bin. For a RUN HOLD job, at the normal termination of the program, two files will be sent to your virtual reader, the status of which can be queried by typing:

```
q r
```

The resulting display may be something like:

```
ORIGINID  FILE  CLASS  RECORDS  CPY  HOLD  FORM
BATCH02   5278  A CON  0000031  001  NONE  STANDARD
BATCH02   5281  A PRT  0000249  001  NONE  STANDARD
```

The PRT file contains the output of your batch job and can be viewed by using the Wylbur FETch command

```
wyl fet xxxx
```

where xxxx is the reader file number (5281 in the above listing). To print an output file held in your reader, you could TRANsfer it to your virtual printer:

```
tran r xxxx to * p
```

Where again, xxxx is the reader file number. Alternatively, you may use the PRS exec on PUBRL 192 to print a reader file:

```
prs xxxx
```

Occaisionally, something goes wrong and the batch job is not executed successfully, a condition which is indicated by a non-zero RETURN CODE from batch. The most common cause of failure is allocation of insufficient time (via incorrect choise of JOB CLASS) to complete program execution. In this case the job is "killed" by BMON and the partially written output files are "DISK DUMPED" to your virtual reader, appearing there as class Q PUN files. Although you can FETch these files, their format makes them unsuitable for printing and/or terminal viewing. To restore such files to regular print format type the following lines:

```
order r xxxx   [xxxx is the desired reader file number]
disk load
```

This will save the output file on your A disk and it could subsequently be printed (with cc) or USEd by Wylbur. To save the file on a disk other than A, (B for example) type:

```
swap a b
order r xxxx
disk load
swap a b
```

Finally, you can use the PURge command to eliminate any or all of the files in your reader queue:

```
pur r xxxx      [purge reader file number xxxx]

pur r all       [purge all of the reader files]
```

HINTS AND SUGGESTIONS

- It is possible to call the GO exec direclty with the name of the desired program (or any abbreviation of it) thus bypassing the menu prompt of the exec. For example, to run the MAGIC program, you could either type GO and then input 3 in response to the menu prompt, or quivalently, type either of the following lines:

```
go magic
go mag
go ma
```

However, to preserve uniquness among the programs with identical initials (eg. PATRICIA and PETROS), it is recommended that the program names be abbreviated to no less than 2 letters.

- The execs USE and FETch on PUBRL 192 obviate the need to enter WYLbur before issuing either of these commands. You could simply type while in CMS:

  ```
  use magic input b
  ```

  This will cause the Wylbur editor to be called and a copy of MAGIC INPUT B to be placed in your active file.

- Both the GO exec and SLAC's BATCH exec expect the user to have some space availble on his/her A disk for storage of a few temporary files. It is thus recommended that the A disk be filled to no more than 90% of its capacity; to find out about the available space on your A disk, type:

  ```
  q disk a
  ```

EXAMPLES

Following are examples of using the GO exec for interactive and batch execution of some programs (see page 2 for a listing of notetional conventons).

[Using GO for interactive execution of a program.]

```
R;
use magic input c              [place data into Wylbur active file]
?
go                                   [call the GO exec]

    PROGRAM MENU:

            1 . . . CELL
            2 . . . COLLIDER
            3 . . . MAGIC
```

```
           4 . . . PATRICIA
           5 . . . PETROS
           6 . . . TRANSPORT
           7 . . . TURTLE
           8 . . . DUMMY1
           9 . . . DUMMY2

    INPUT NUMBER FOR THE DESIRED PROGRAM, OR <CR> TO EXIT:

3                                         [select the MAGIC program]
IS THIS A BATCH OR AN INTERACTIVE JOB?
B=BATCH, <CR>=INTERACTIVE.
                                          [<CR> for interactive job]

SPECIFY DISK TO SAVE OUTPUT FILE ON (<CR>= B)
                                          [<CR> for b disk]
R;
EXECUTION BEGINS...
$.
$.
R;
SLAC VM-WYLBUR (VERSION OF 03/17/82 - DCSS)
EXEC END
   53.     0 ***CONVERGENCE: FUNCTION=0.279890D-05 & FIT=-.155944D-08***
THICK-LENS SOLUTION WITH  35 ITERATIONS
  237.        CONV.  .3D-05               [at the completion, output is
?                                          placed in Wylbur and the command
qq                                         I 'conv' is issued]
R;
l magic * b                               [verify that input and output
MAGIC    INPUT    B1                       files have been saved on the
MAGIC    OUT06    B1                       selected (b) disk]
R;
print magic out06 b (cc                  [obtain a printed copy of output]
PRT FILE 2168  FOR HXSRL    COPY 001
R;
```

<u>[Using GO directly with the name of the desired program]</u>

```
use magic input c
?
go me                                     [the menu prompt is bypassed]
IS THIS A BATCH OR AN INTERACTIVE JOB?
B=BATCH, <CR>=INTERACTIVE.

                                          [input <CR> for interactive job]
SPECIFY DISK TO SAVE OUTPUT FILE ON (<CR>= B)
                                          [input <CR> for b disk]
R;
```

```
EXECUTION BEGINS...
$.
$.
R;
SLAC VM-WYLBUR (VERSION OF 03/17/82 - DCSS)
EXEC END
   53.      0 ***CONVERGENCE: FUNCTION=0.279890D-05 & FIT=-.155944D-08***
THICK-LENS SOLUTION WITH  35 ITERATIONS
  237.         CONV.  .3D-05             [output is placed in Wylbur active
?                                         file at the completion of the job]
```

<u>[Using GO to execute a batch job]</u>

```
use magic input c                       [place data into Wylbur active file]
?
go                                      [call the GO exec]

    PROGRAM MENU:

           1 . . . CELL
           2 . . . COLLIDER
           3 . . . MAGIC
           4 . . . PATRICIA
           5 . . . PETROS
           6 . . . TRANSPORT
           7 . . . TURTLE
           8 . . . DUMMY1
           9 . . . DUMMY2

    INPUT NUMBER FOR THE DESIRED PROGRAM, OR <CR> TO EXIT:

3                                         [select the MAGIC program]

IS THIS A BATCH OR AN INTERACTIVE JOB?
B=BATCH, <CR>=INTERACTIVE.
b                                         [input b for batch]

SPECIFY DISK TO SAVE INPUT FILE ON (<CR>= B)

                                          [<CR> for b disk]
RUN HOLD? YES, OR <CR> FOR RUN
y                                         [y for RUN HOLD; the output will
                                           be sent to my reader]

SPECIFY JOB CLASS: (TIME LIMITS ARE IN CPU MINUTES)

X=1; S=2; B=4; M=8; L=30; J=120; (<CR>=X).
```

```
                                             [<CR> for class x (1 minute job)]

R;
PUN FILE 2211  TO  BMON     COPY 001   NOHOLD

Job HXS147 submitted.                        [jobname is hxs147]
R;
R;
JOB HXSRL HXS147 QUEUED TO RUN 04/30 16:14:33, CLASS X , PRTY 09, POS 1

jobstat I                                    [using JOBSTAT to locate the job]
$.
HXSRL  HXS147  QUEUED TO RUN  04/30/82 16:14:33 class X  60 sec PRTY  9
R;

FROM BATCH01 :  JOB " HXS147 " ENDING.. IN BATCH01 AT 16:15:19
FROM BATCH01 :  BATCH RETURN CODE = 0
FROM BATCH01 :  VTIME=000:04 TTIME=000:05 SIO=000441
FROM BATCH01 :  RDR-000442 PRT-000308 PCH-000442
CON FILE 2215 FROM BATCH01  COPY 001   NOHOLD
RDR FILE 2221 TRANSFERRED FROM BATCH01
                                          [job completion messages from batch]

q r                                       [query the files sent to your reader]

ORIGINID FILE CLASS RECORDS  CPY HOLD FORM
BATCH01  2215 A CON 00000031 001 NONE STANDARD
BATCH01  2221 A PRT 00000249 001 NONE STANDARD  [this is the output file]
R;
fet 2221                              [use the Wylbur FET command to examine
                                       the output file queued in your reader]
?
l 'conv'
   53.      ***CONVERGENCE: FUNCTION=0.279890D-05 & FIT=-.155944D-08***
THICK-LENS SOLUTION WITH  35 ITERATIONS
  237.       CONV.  .3D-05
?
qq                                    [exit wylbur]
R;
tran r 2221 to * p                    [transfer output file to your printer
                                        to get a printed copy]
PRT FILE 2221 TRANSFERRED FROM *        RDR
0001 FILE  TRANSFERRED
R;
q p                                   [query the files in your printer]
ORIGINID FILE CLASS RECORDS  CPY HOLD FORM
BATCH01  2221 A PRT 00000249 001 NONE STANDARD   [output file in printer]
R;
```

[An example of using JOBSTAT and BATCH CANCEL]

```
use magic input c                    [place data into Wylbur active file]
?
go mag                                    [select MAGIC program]
IS THIS A BATCH OR AN INTERACTIVE JOB?
B=BATCH, <CR>=INTERACTIVE.
b                                         [b for BATCH job]

SPECIFY DISK TO SAVE INPUT FILE ON (<CR>= B)  [select b disk]

RUN HOLD? YES, OR <CR> FOR RUN
y                                              [run hold]

SPECIFY JOB CLASS: (TIME LIMITS ARE IN CPU MINUTES)

X=1; S=2; B=4; M=8; L=30; J=120; (<CR>=X).
1                                           [class 1 (30 minutes)]
R;
PUN FILE 2317  TO  BMON     COPY 001   NOHOLD
Job HXS148 submitted.                       [job is submitted]
R;
R;
JOB HXSRL HXS148 QUEUED TO RUN 04/30 16:20:11, CLASS L , PRTY 09, POS 24

jobstat 1                                   [locate the job]
HXSRL HXS148  QUEUED TO RUN  04/30/82 16:20:11 class L 1800 sec PRTY 9
R;
batch cancel hxs148                         [cancel the job]
PUN FILE 2331  TO  BMON     COPY 001   NOHOLD
R;
FROM BMON    :  JOB HXSRL HXS148 HXSRL CANCEL 04/30/82 16:20:48
jobstat 1                                   [locate job]
NO JOBS                                     [job has been cacelled]
R;
```

Appendix A

OBTAINING TEMPORARY DISK SPACE

At times it is convenient (or even necessaery) to have access to a temporary disk to store intermediate output files before they are either printed or further processed by other programs. The command to obtain a temporary disk of n cylinders is:

```
gime n
```

The system will respond in a few seconds with the message

```
xxxmmm (mmm D)-- n CYL 3380 TDISK
```

```
Where:    xxx is your account name (not VM id)
          mmm is the virtual disk address
          D   is the disk mode; it may be a different letter
              depending on how many disks you have.
```

A disk of 5 cylinders would provide sufficient storage for the output of any program available through the GO exec.

It should be noted that whenever you LOGoff, your temporary disks disappear and the files stored on them are irretrievably lost. DISConnecting ("disc") however, instead of LOGing off will ensure that you will retain access to your temporary disks for at least 24 hours (system crashes not withstanding!)

Appendix B

ADDING NEW PROGRAMS TO THE GO EXEC

Provisions have been made to allow addition of new programs to the GO exec's program list by simply changing either of the two names "DUMMY1" or "DUMMY2" to the name of the new program. As the 'GO Package' consists of two execs, GO and GOSUBS, the changes have to be made to both of these execs. Only standard input/output file definitions have been provided for the DUMMY1 and DUMMY2 programs: input from logical unit 5 and output to logical unit 6. Other variations and/or additons could easily be implemented by modifying the sections of the GOSUSS exec titled DUMMY1 and DUMMY2.

Thus users can tailor the GO exec to their individual computing needs by modifying the 'master' copy on PUBRL 192 and saving the 'personalized' version on one of their own disks. Additionally, the users should provide appropriate TEXT files containing object modules of the new programs (to find out how to compile a program, type: "help forthx (all"). It should be noted that in batch mode only the execs and programs on PUBRL 192 are accessed; hence, these newly added programs can only be run interactively, unless both the modified versions of the execs and the new TEXT files are saved on PUBRL 192. Alternatively, the BATCH SUBMIT command of the GO exec maybe changed to allow access to your disks.