Edinburgh Regional Computing Centre

User Note 53

(May 1985)

Title:

Background Work on EMAS 2900

Author: Neil Hamilton-Smith et al.

Contact: Advisory service

Software Support Category: See Note 15

Synopsis

This Note gives a simple description of foreground and background EMAS 2900 process concurrency. This is followed by descriptions of some ways of using a background process.

Keywords

B80, BFORT77, BFORTE, BIMP, BIMP80, BINTT, BIOPT, BOP, BPASCAL, DELAYJOB, DESCRIBE, DETA, DETACH, DETACHJOB, process concurrency, S£CERRS, S£JJN, S£LIST, SS£DET

Introduction

Most of the commands described in this note are automatically accessible to all users. Four of them (B80, BOP, DESCRIBE and DELAYJOB) are not accessible until you have given the command

```
OPTION SEARCHDIR=CONLIB.GENERAL
```

You only need to issue this command once.

On-line information about each command may be found by using the HELP command; where printed documentation exists a reference is given in the description.

Process Concurrency

EMAS 2900 is designed to allow many processes belonging to a given username to run concurrently, and by default a limit of one interactive and one simultaneous batch process is allowed.

Users can find their concurrency limits by using the command DESCRIBE which is accessed via CONLIB.GENERAL. For example,

```
Command:DESCRIBE
*EKLD91*     Fsys: 10
            Owner: J.Hunter Dunn      Delivery: Aldershot
         Basefile:
  Privilege(ACR): 10 (most privileged=1, usual/students=10, least=15)
       Processes: Running 1 (fore), 0 (back)
                  Maximum 1 (fore), 1 (back), 2 (concurrent)
   Passwords set: 24/08/85 at 16.26.07 (fore), 01/08/85 at 15.02.44 (back)
   Last logged on: 25/08/85 at 16.38.26 (fore), 17/08/85 at 17.29.16 (back)
       Funds left: 142.00  (Group holder: EKLD01)
-------------------------------------------------------------------------------
    FILE INDEX: 5 Kb, 66 files
         Files: Currently 51 permanent (1664 Kb), 15 temporary (2148 Kb)
   Descriptors: Used(free)   Files 66(27)  Sections 17(31)  Permissions 1(31)
        Limits: Total 2048 Kb,  per file 1024 Kb
-------------------------------------------------------------------------------
ARCHIVE INDEX: 20 Kb (£ARCH)
         Files: 30 Archive (1000 Kb), 219 Backup (26572 Kb)
   Descriptors: Used(free)   Files 249(369)  Permissions 0(64)
-------------------------------------------------------------------------------

Command:
```

A user may ask the System Manager to change his process concurrency limits. This can best be requested by MAIL to SYSMAN.

Batch Jobs

The batch streams in EMAS 2900 may be used for work involving moderate or large amounts of processor time, where one would expect to see results at the terminal only after a long elapsed time. The DETACH mechanism for batch jobs provides a means of spreading the computing

load into periods when the system is less heavily loaded (see Chapter 16 of the EMAS 2900 User's Guide). The limits which apply to the batch streams mean that batch jobs requiring large amounts of processor time will tend to wait until off-peak periods before execution commences (see Appendix 5 of the EMAS 2900 User's Guide). The numbers of batch streams and their cpu limits are adjusted to give as reasonable a turn-round as is possible.

Users who specifically require batch execution to take place when they are not active in a foreground process can use the "AFTER=" parameter in the DETACH command. The following exemplifies use of the "AFTER=" document parameter to set a time before which the job cannot commence:

*Command:*DETACH JOBFIL,30,.IN
*DOC param:*AFTER=18.00.00
DOC param:.END

JOBFIL is Batch queue entry 835. Time limit 30 secs.

By using two processes, one interactive and the other batch, it is possible to have compilations, and indeed any other work, performed 'in the background', while other, perhaps unrelated, work continues in the foreground process.

One effect of concurrency is that users will need to be aware of the possible effects of attempting to connect a file in write-mode in one of the processes when the other process has the file connected (see Chapter 1 of the EMAS 2900 User's Guide); but accepting the possible inconvenience which may result, it will be possible to have a non-interactive computation going on in the background while the user continues interactive work at the terminal. If a background and a foreground process both need to read the same file at the same time, they can both do so. If one of them wants to write to the file, only the 'first comer' can use the file. The other process will find its attempts to get at the file are rejected. EMAS 2900 software is arranged so that this does not normally occur by accident, but it can easily be provoked, for instance by attempting to edit a file in the foreground while the same file is being compiled in a batch job. No deep understanding of EMAS 2900 is needed to avoid these situations. In those few cases where the restriction is a real problem, the command NEWGEN will often provide the solution. NEWGEN is a command to allow a new version of a file to be superimposed upon an existing file even though it is currently in use by other processes (see Chapter 5 of the EMAS 2900 User's Guide).

Beware, however, of using NEWGEN without due thought. If you have a batch job which sometimes "fails", you may be able to get it to "work" by using NEWGEN. But sometimes the "failure" is a warning that you are trying to make simultaneous changes to a file from two concurrent jobs. Using NEWGEN simply to suppress the warning can lead to surprising difficulties, such as changes made to a file being "lost" in later versions of the file.

Job Control Language

If something goes wrong in a batch job, there is no opportunity for the user to decide what to do to put things right as one might do in interactive working. For instance, if a compilation fails, the next command will be obeyed even if that is an attempt to RUN the compiled program. The EMAS 2900 job control language (which is distinct from the interactive command language) is available for use in batch jobs and provides facilities to cope with these problems. DETACHJOB is the command to use if you want a batch job to use the job control language. For interactive use, OBEYJOB is available. The Job Control Language is documented in Chapter 16 of the EMAS 2900 User's Guide.

Except where explicitly stated, all the facilities in the Note use the basic command language, not the batch Job Control Language.

Commands for Background Compilation

To take advantage of process concurrency, a set of Subsystem commands is offered to allow easy detaching of compilations for FORT77, FORTE, IMP, IMP80, IOPT (see Appendix), and PASCAL. The commands are respectively BFORT77, BFORTE, BIMP, BIMP80, BIOPT and BPASCAL, and the parameters are similar to those for the corresponding foreground compilations.
For example:

```
BFORT77 source,object/.null,compilation listing/.null,errors,time
```

where

source	is the name of the file containing the source statements.
object	is the name of the file generated by the compiler. This file cannot be a member of a partitioned file.
compilation listing	is the name of the file or device onto which the compiler will write its listing. It is not possible to specify a member of a partitioned file as this parameter. By default this parameter assumes the file named S£LIST.
errors	is the name of the file or device onto which the compiler will write error messages. It is not possible to specify a member of a partitioned file as this parameter. By default this parameter assumes the file named S£CERRS and destroys this file if no errors occur.
time	is an upper limit on the number of seconds to be allowed for the compilation. The command then adds 5 seconds to this figure to allow for the other activities which are described below. By default the time parameter is set to a value 5 greater than some figure calculated from the size of the file of source statements, and so is never less than 6 seconds.

The PARM setting obtaining at the time of the BFORT77 (etc.) command is used for the compilation, and a batch job is detached with a time limit which is either the default described above or the value specified by the fifth parameter. The detached job prepares the object and listing in temporary files which are NEWGENned to the corresponding named files (if already existing) when the compilation is complete. By using this mechanism the detached job allows the old object and listing files to be used in the foreground (interactive) process without let or hindrance even while the background compilation is proceeding. In this case it will be necessary to DISCONNECT the object and listing files in the foreground process after the compilation is complete (or DISCONNECT .ALL if appropriate) in order to gain access to the new versions. The old object file will *not* however, be replaced if the compilation does not succeed, to avoid unnecessarily destroying a valid object file.

The background job sends messages to the foreground process on completion to report success or failure of the compilation and the name of a file containing the journal for the detached job in the case of failure. This file is called S£JJN. It should not normally be necessary to inspect this latter file, though it may be useful in the event of abnormal termination of the background job.

The following is an example of part of a terminal session in which BFORT77 is invoked.

*Command:*PARM OPT

*Command:*BFORT77 F1SRCE,TRAINY,L

T£DETACH is Batch queue entry 855. Time limit 6 secs.

*Command:*EDIT other file
..
..
etc.

***EKLD91 17/03/85 10.57:*
Compilation BFORT77 F1SRCE,TRAINY,L successful
***EKLD91 17/03/85 10.57: BATCH JOB T£DETACH COMPLETED*

or alternatively, the output might have been:

***EKLD91 17/03/85 10.57:*
BFORT77 F1SRCE,TRAINY,L failed -
A fault listing has been sent to S£CERRS and file S£JJN
***EKLD91 17/03/85 10.57: BATCH JOB T£DETACH COMPLETED*

Caution

It is inadvisable to edit the source file before the batch job is complete as you do not know whether the compiler will compile the old or the new source text. In some cases the compilation may even fail if the editor has the file connected in write-mode at the moment when the compiler attempts to access the source. Correspondingly some editors may not work if they cannot access the file in write-mode because the background job is executing.

BFORT77, BIMP80 etc. are designed to take the same parameters as the corresponding foreground compilation commands FORT77, IMP80 and so on. They behave as nearly as possible in the same way as the foreground commands.

There are two other commands which make it easy to detach compilations of IMP programs. These commands, B80 and BOP, are accessed via CONLIB.GENERAL. B80 detaches a job to call the IMP80 compiler, and BOP detaches a job to call the IOPT compiler. B80 and BOP do not emulate the behaviour of IMP80 and IOPT, but they offer extra features which may be found convenient.

The form of these commands is

*Command:*B80 source, parm, parm, ...

where source is the name of the file containing the source statements: it is the first, and only necessary parameter. If the filename is FILE the object and listing files will be called FILEY and FILEL. If the filename ends with "S", the "S" is dropped before the "Y" and "L" are appended. The filename must have less than 11 characters unless the last is "S". If the filename contains a 6-character username or is a partitioned file membername, the object and listing filenames are generated from the filename or membername.

The following keywords may be appended, separated by commas, as additional parameters to the command:

.NY	object file to be .NULL.
.N	listing file to be .NULL (also adds PARM NOLIST).
.OUT	error lines to interactive terminal.
.LP	listing file copied to .LP.
.LPD	(".LP and Destroy") listing file sent to .LP.
X	object filename to have X rather than Y (or Z) as last character.
NEWGEN	object file to have X as last character and, provided compilation is successful, object file is NEWGENned onto existing "Y" (or "Z") object file.

The following all cause the corresponding PARM to be set. When compilation is complete PARM is reset to the parms which previously obtained.

NOCHECK	By default a check is made that each variable whose value is about to be used has been assigned: this is inhibited by NOCHECK.
NODIAG	By default symbol tables are retained at run-time, for the production of values in local variables in diagnostics: this is inhibited by NODIAG.
NOLINE	By default code is included to keep a record for diagnostics of the number of the line being executed: NOLINE inhibits this.
NOLIST	By default a program listing is generated at compile-time. NOLIST causes output of only a heading and any error messages.
OPT	This - optimise code - gives rather more than NOCHECK, NOLINE, NODIAG (for example, stack overflow is not checked).
NOTRACE	By default routine traceback information for diagnostics is kept: this is inhibited by NOTRACE.

NOARRAY By default an array bound check is carried out on all array accesses: this is inhibited by NOARRAY. Note that some checking on multidimensional array bounds will be carried out even if NOARRAY is selected.

DEBUG Related to the IMP 'debug' facility: see User Note 8, Developing IMP Programs.

PROFILE This causes extra code to be inserted to keep a count of how often each line is executed. Call of routine PPROFILE then gives summary of information and resets counts. PARM NOLINE is forced off.

DYNAMIC This marks all references for satisfying dynamically. In addition, %EXTERNALROUTINEs are loaded dynamically if they are specified in the calling program as, for example, %DYNAMICROUTINESPEC. See also "Dynamic loading", in Chapter 11 of the EMAS 2900 User's Guide.

STACK This causes arrays to be put on the user stack, rather than on an auxiliary stack; liable to fail with "stack overflow". For special cases only. (See "The use of the stack", in Chapter 11 of the EMAS 2900 User's Guide and User Note 33.)

MAXDICT This causes the compiler tables to be set to the maximum possible size.

The following examples show part of a terminal session in which B80 is invoked.

*Command:*B80 IS1S
Parms set: DEFAULTS
T£DETACH is Batch queue entry 842. Time limit 6 secs.

***ERCN19 28/03/85 12.50: Compilation BIMP80 IS1S,IS1Y,IS1L successful*
***ERCN19 28/03/85 12.50: BATCH JOB T£DETACH COMPLETED*

*Command:*B80 IS1S,.LP,NOARRAY,NOCHECK
Parms set: NOCHECK,NOARRAY
T£DETACH is Batch queue entry 843. Time limit 6 secs.

***ERCN19 28/03/85 12.51: Compilation BIMP80 IS1S,IS1Y,IS1L successful*
***ERCN19 28/03/85 12.51: BATCH JOB T£DETACH COMPLETED*

*Command:*B80 MYIMPPROGS_S1,.NY
Parms set: DEFAULTS
T£DETACH is Batch queue entry 845. Time limit 6 secs.

***ERCN19 28/03/85 12.53:*
Compilation BIMP80 MYIMPPROGS'S1,.NULL,S1L successful
***ERCN19 28/03/85 12.53: BATCH JOB T£DETACH COMPLETED*

Utility DETA

If you require a background job to be processed by the command interpreter and the job text already exists in a file then use the Subsystem command DETACH. If such a file does not exist then the command DETA, which is accessed via CONLIB.GENERAL, could be used. The DETA command is given without parameters. In response to the prompt *Data:* you type first the commands, then the scheduling parameters (if any), and finally a time limit or terminator. For example:

*Command:*DETA
*Data:*PARM OPT
*Data:*FORT77 F1SRCE,TRAINY,L
Data:

Optional scheduling parameters may now be provided. The timing of the execution of the job depends upon the scheduling parameters specified (see below) and on the job scheduling rules operated by the Service Manager; see Chapter 16 and Appendix 5 of the EMAS 2900 User's Guide. A full list of scheduling parameters is given in Table 2.1 of the EMAS 2900 User's Guide: the following is a useful subset of them.

Parameter	Meaning
OUT	The destination of the journal produced for the job. Possible destinations are: FILE NEWFILE NULL and any valid output device, for example LP (which is the default), or LP23.
OUTNAME	If the destination specified for OUT is FILE or NEWFILE then OUTNAME must be used to specify a valid filename by which the journal file will be identified in the user's process. If OUT=NEWFILE is specified then a file of this name must not already exist: if OUT= is not specified then NEWFILE will be assumed.
RERUN	If the document is a job, whether the job is to be rerun if caught in a machine crash. If the document is an output file, whether it should be reprinted if caught in a machine crash. Should be specified as YES or NO.
AFTER	The document is to be held in the queue until after the date and time specified. Valid values are of the form: date time time date date time where "date" is in the form dd/mm/yy, and "time" is in the form hh.mm.ss.

If any of these scheduling parameters are used, each must be followed by an equals sign and the value to be assigned to it. Each such statement must be on a line by itself. These statements may either be given separately in response to the *Data:* prompt, or edited into a parameter file named PF which DETA will automatically use. This can be readily used (for instance) to suppress the job-listing which is sent to .LP, by diverting it to a file (say) JUNK. For example, the contents of file PF to do this would be:

```
OUT=FILE
OUTNAME=JUNK
.END
```

Finally either an optional CPU time limit for the *whole* job or a terminator must be supplied. The time is given as one or two integers:

integer	minutes
,integer	seconds
integer,integer	minutes and seconds.

This information is used for scheduling purposes, as well as imposing a limit on the time used for the job. Refer to Appendix 5 of the EMAS 2900 User's Guide for details of the default and maximum values for this parameter.

Once a time has been specified the job is detached for execution, and control returns to command level. For example,

*Command:*DETA
*Data:*PARM OPT
*Data:*FORT77 F1SRCE,TRAINY,L
Data:,42
SS£DET is Batch queue entry 412. Time limit 42 secs.

Command:

If the default time is deemed to suffice there are three alternative ways of indicating that all the necessary information for the job has been given:

%C	detaches the job and returns to command level.
:	has exactly the same effect as %C.
Q	returns to command level without detaching the job file.

The first example below shows how the job can be put in the background job queue, with all output being directed to the local line printer (the file PF described above is assumed not to exist).

*Command:*DETA
*Data:*PARM OPT
*Data:*FORT77 F1SRCE,TRAINY,L
Data:%C
SS£DET is Batch queue entry 413. Time limit 30 secs.

If you want to detach several jobs, successive job-listings can be sent to files having names J1, J2, ... by calling (once only) DETA INIT. At this call a PROFILE entry is created in which the last-used filename is remembered (for details of the PROFILE scheme see User Note 16). The series of filenames restarts at J1 each day on which DETA is so used. If you want to use this facility, file PF must not exist. This facility may be disabled by calling DETA CANCEL. For example,

*Command:*DETA INIT

*Command:*DETA
*Data:*PARM OPT
*Data:*FORT77 F1SRCE,TRAINY,L
Data:%C
SS£DET is Batch queue entry 579. Time limit 30 secs.
Output file J1

*Command:*DETA
*Data:*DEFINE 6,.LP15
*Data:*RUN TRAINY
Data:%C
SS£DET is Batch queue entry 580. Time limit 90 secs.
Output file J2

Utility DELAYJOB

The command DELAYJOB can be used within a background job to cause the job to rerun itself at regular intervals, and may be found useful for accounting, monitoring, mail systems, etc. It is accessed via CONLIB.GENERAL.

The required parameters are, with the exception of the first, similar to those of DETACHJOB:

```
DELAYJOB daysdelay,jobfile,cputime,paramfile
```

daysdelay - a positive integer specifying the number of days delay after which the job is to be rerun.

jobfile - a sequence of commands and data which could have been presented in a foreground job. The last two commands must be

```
DELAYJOB daysdelay,jobfile,cputime,paramfile
.ENDJOB
```

cputime - the cpu time limit for the whole job.

paramfile - an existing character file of scheduling parameters suitable for use with the EMAS 2900 DETACHJOB command, i.e. to be interpreted by the EMAS 2900 Job Interpreter. In this instance, 'paramfile' must contain (at least):

```
AFTER=dd/mm/yy
.END
```

to specify the first date on which the job is to run. Any 'time of day' parameter specified must come after the date string.

Each time the DELAYJOB command executes, it rewrites the date within 'paramfile' to advance it. The increment will always be as specified, irrespective of the current real-world date. Any other parameters, such as 'time of day', optionally included in 'paramfile', are not altered. After the date is rewritten, 'jobfile' is detached automatically, i.e. DETACHJOB jobfile,cputime,paramfile is executed.

Example

To set up a jobfile 'JOBFIL' to rerun automatically every 7 days, with 35 seconds cpu time allowed for the job.

The jobfile JOBFIL could be edited to include the following commands:

```
DELAYJOB 7,JOBFIL,35,PARFIL
DEFINE 6,.LP15
RUN TRAINY
.ENDJOB
```

This will enable the job to detach itself, to be run at a later date.

A character file, 'PARFIL' say, could be created containing:

```
RERUN=YES
OUT=.LP15
AFTER=27/03/85
.END
```

This specifies the peripheral details for all runs and the date of the first run of TRAINY, 27/03/85.

The *first* run of the job may then be initiated by:

Command: DETACHJOB JOBFIL,35,PARFIL
JOBFIL is Batch queue entry 942. Time limit 35 secs.

Once set up in this way, a job can only be cancelled by the DELETEDOC command. The onus is on the user to appreciate how the 'AFTER=' facility operates, and to allow for machine shut-downs, queueing delays or job failures.

A job set up to run every Monday, say, should always execute on a Monday unless the machine is not in service, in which case the job will run as soon as service is restored - but the next job will be scheduled correctly for the following Monday. There is a necessary safeguard built into the DELAYJOB routine, which will detect a situation where the date specified is so old that the job would have to be rerun more than 5 times today to catch up. The run aborts to prevent an avalanche of jobs if the date was wrongly specified - it will not affect the normal running of the job except in the case of exceptionally long machine down-times.

Interrogating Background Jobs from Foreground - BINTT

The command BINTT (Background INT:T) gives an interactive user a means of interrogating the user's own background jobs while they are running; BINTT takes no parameters.

The information returned by BINTT is similar to, but expands upon, the type of information returned by a foreground *Int:T*. In addition to time used and page turns, BINTT identifies the interrogated job's document number, invocation number, job file and the command within the job file which is currently being executed. The invocation number is used to distinguish between multiple concurrent processes for the same username. Note that, as in *Int:T*, the time and page turns returned are those for the command in execution, not for the job as a whole.

The command works by first establishing whether any background jobs belonging to the caller are currently running. If there are any then an interrupt is sent to each one requesting status. The background process stops what it is doing, collects together job status information then sends a TELL message to the foreground process before resuming where it was interrupted. There are two standard replies from BINTT:

1. *Command:*BINTT
   ** No background jobs found **

2. *Command:*BINTT
1 background job(s) interrogated

***EKLD91 17/03/85 10.56:*
Jobname=T£DETACH Doc=0855: T=4 PT=21 Invoc=1
Comm=FORT77 F1SRCE,TRAINY,L

Additionally two other cases may occasionally arise.
The first is when the current command line says just Comm=, thus:

***ERCI19 17/03/85 16.59:*
Jobname=T£DETACH Doc=0956: T=1 PT=251 Invoc=1
Comm=

Command:

Here the interrupt was accepted but arrived so early in the creation of the background process that it had not reached the stage of executing a user command.

The second case is when BINTT confirms that a background job has been interrogated but no reply is received: the interrupt has been ignored.

*Command:*BINTT
1 background job(s) interrogated

Command:

This can happen if the interrupt arrives just as the background process is starting up or stopping. In the former case another BINTT should be successful, in the latter the standard 'Batch job completed' message should follow almost immediately.

Appendix

The IOPT IMP80 Compiler

Development of the IMP80 compiler for EMAS-3 has been under way for some months. The new compiler will have a language specification as close as possible to the present IMP80 - in particular long integers not supported by IBM hardware will be provided by software. User Note 58 gives further information about moving IMP programs to EMAS-3.
The first three passes of the new compiler have been married with the current IMP80 code generator for testing purposes and this compiler is available on the 2976 (host name EMAS) under the command IOPT.
When invoked with PARM OPT set IOPT performs a number of optimizations not attempted by the current IMP80 and can result in noticeable improvements in run time speed. However, certain 2900 specific optimizations that are included in IMP80 but are not relevant to IBM architecture have been omitted so performance gains are not inevitable.

IOPT modules can be linked freely with modules compiled by IMP80 and parameter passing should be completely compatible. There should be no assembler in modules presented to IOPT.

Users are invited to try IOPT and report any problems by EMAS mail to 'P.Stephens'.