

ABORTJOB.COM

```
⌘ ! abortJob.com
⌘ ! this file is used to abort a batch job
⌘ ! p1 is the jobNumber; p2 is the queue
⌘ ! The "show batch" command is used to determine if the job
exists. If
⌘ ! it does not, the message NIL is returned; otherwise, the
job is
⌘ ! aborted.
⌘ ! All messages are returned to the user's root directory.
⌘ ! If there is a serious error, ...
⌘ ! if there is an error in the running of this com file, the
detailed
⌘ ! error message gets sent to abortJob.err in the user's root
directory.
```

```
⌘ !SET VERIFY
⌘     delete sys⌘login:abortJob.err.*
⌘     delete sys⌘login:abortJob.res.*
⌘ SET NOVERIFY
⌘     define sys⌘output abortJob.tmp
⌘     show queue 'P2'
⌘     deassign sys⌘output
⌘ !SET VERIFY
⌘     open/write result sys⌘login:abortJob.res
⌘
⌘ loop:
⌘     open/read file abortJob.tmp
⌘     read/end_of_file=done file line
⌘     jobNumber = f⌘integer(f⌘extract(32,4,line))
⌘     if jobNumber .eq. P1 then goto found
⌘     goto loop
⌘
⌘ done:
⌘     write result "( OK NIL)" ! job not found
⌘     goto finish
⌘
⌘ found:
⌘     define sys⌘error sys⌘login:abortJob.err
⌘     on error then goto error
⌘     stop/entry='P1' 'P2'
⌘     deassign sys⌘error
⌘     write result "( OK ( Job ",P1," on queue ",P2,-
" has been aborted))"
⌘
⌘ finish:
⌘     close result
⌘     close file
⌘     delete abortJob.tmp.*
⌘     exit
```

```

x error:
x     @[gslws.server]error sys@login:abortJob.res 'xSTATUS'
x     deassign sysxerror
x     close result
x     close file
x     delete abortJob.tmp.*
-----

```

COMPILE.COM

```

x !  COMPILE.COM      8/7/86
x ! this file is used to compile a job interactively.
x !  job is the name of the user's fortran source file
x ! the file exists in the user's local directory, which may be a
x ! subdirectory of the root directory.
x ! the object file is made in the user's local directory.
x ! if there is no error in compilation, the name and date of the
x ! object file are returned in sys@login:compile.res.
x ! if there is an error in compilation, the abbreviated error
message
x ! is returned in sys@login:compile.res, and the detailed
x ! error message is written to sys@login:compile.err.
x
x
x !SET VERIFY
x     job = fxparse("'P1'",,, "name")
x     userDirectory = fxparse("'P1'",,, "directory")
x     length=fxlength(job)
x !     show symbol job
x !     show symbol userDirectory
x !     show sym length
x     delete sys@login:compile.err.*
x     delete sys@login:compile.res.*
x     delete 'P1'.obj.*
x     define sysxerror sys@login:compile.err
x !     define sysxerror sys@login:'job'.err
x     on error then goto error
x
x     fortran/object='userDirectory''job' 'P1'
x     deassign sysxerror
x SET NOVERIFY
x     define sysxoutput sys@login:objFile.tmp
x     dir/date 'P1'.obj
x     deassign sysxoutput
x !SET VERIFY
x     open/write resultFile sys@login:compile.res
x     open/read file sys@login:objFile.tmp
x
x loop:
x     read/end_of_file=done file line
x !     show sym line
x     name=fxextract(0,length,line)
x !     show sym name

```

```

*      if name .eqs. job then goto found
*      goto loop
*
* done:
*      write resultFile "( OK NIL)"      ! object file not found
*      goto finish
*
* found:
*      write resultFile "( OK (",line,"))"
*
* finish:
*      close resultFile
*      close file
*      delete objFile.tmp.*
*      exit
*
* error:
*      @[gslws.server]error sys#login:compile.res '#STATUS'
*      deassign sys#error
-----

```

ERROR.COM

```

* ! lists error status message in specified file
* ! call by: @error resultFile errorStatus
*
*      open/write result 'P1'
*      errorFile = f#logical("sys#error")
* !      show sym errorFile
*      shortName=f#parse(errorFile,,,"name")
* !      show sym shortName
*      write result "(ERROR ""' f#message(P2)' ""
*'shortName'.ERR)"
*      close result
-----

```

LINK.COM

```

* ! LINK.COM      8/8/86
* ! This file is used to link a series of object files to form an
* ! executable file.
* ! The parameter P1 is the object code filename of the main
file.
* ! The parameter P2 is a string composed of all object files to
* ! be linked with P1. There must be a comma between these
files
* ! within P2.
* ! Job is the extracted name of the user's main object code
file.
* ! This file exists in the user's local directory, which may
be a
* ! subdirectory of the root directory.
* ! The executable file is made in the user's local directory.
* ! If there is no error in linking, the name and date of the

```

```

* ! executable file are returned in sys%login:link.res.
* ! If there is no error in linking but no .exe file is made, a
* ! message to that effect is returned in sys%login:link.res.
* ! If there is a link warning during linking, an error message
is returned
* ! in sys%login:link.res, and the detailed link warning
messages are
* ! written to sys%login:link.err.
* ! If there is an error in linking, such as no existing object
file,
* ! the abbreviated error message is returned, from the
ERROR.COM file,
* ! in sys%login:link.res, and the detailed error message is
written
* ! to sys%login:link.err.

* !SET VERIFY
*     job = f%parse("'%P1'",,, "name")
*     userDirectory = f%parse("'%P1'",,, "directory")
*     length=f%length(job)
* !     show symbol job
* !     show symbol userDirectory
* !     show symbol length
*     delete sys%login:link.err.*
*     delete sys%login:link.res.*
*     delete 'P1'.exe.*
*     define sys%error sys%login:link.err
* !     define sys%error sys%login:'job'.err
*     on error then goto error
* !
* ! Note: link warnings can be very serious, such as the absence
of object
* ! code modules, in which case a useless .exe file is made.
Because
* ! errors (as opposed to warnings) get trapped through the
error routine,
* ! these serious link warnings must be handled specially.
* !
* !     show symbol P2
*     if P2 .eqs. "" then goto simple
*     link/exe='userDirectory''job' 'P1','P2'
*     goto continuel
* !
* simple:
*     link/exe='userDirectory''job' 'P1'
* !
* continuel:
*     deassign sys%error
*     open/write resultFile sys%login:link.res
* !
* ! If we've gotten this far, it means no errors occurred.
* ! First, check if link warnings occurred, by determining if a

```

```

x ! LINK.ERR file was written. If so, continue through
linkerror1.
x !
x SET NOVERIFY
x     define sys$output sys$login:linkFile.tmp
x     dir/date/siz sys$login:link.err
x     deassign sys$output
x !SET VERIFY
x     open/read file sys$login:linkFile.tmp
x !
x loop1:
x     read/end_of_file=continue2 file line
x     show sym line
x     name=f$extract(0,4,line)
x     show sym name
x     if name .eqs. "LINK" then goto linkerror1
x     goto loop1
x !
x continue2:
x     close file
x !
x ! Second, check if an executable file was made. (Executable
files are
x ! made in spite of link warnings. The following check flags a
a
x ! situation where neither a link warning nor an executable
file is made.)
x !
x SET NOVERIFY
x     define sys$output sys$login:exeFile.tmp
x     dir/date 'P1'.exe
x     deassign sys$output
x !SET VERIFY
x     open/read file sys$login:exeFile.tmp
x !
x loop2:
x     read/end_of_file=linkerror2 file line
x     show sym line
x     name=f$extract(0,length,line)
x     show sym name
x     if name .eqs. job then goto found
x     goto loop2
x !
x linkerror1:
x     message="error during linking"
x     write resultFile "(ERROR "'message'" LINK.ERR)" !
link warning
x     goto finish1
x !
x linkerror2:
x     message="executable file not made"
x     write resultFile "( OK ("message,"))" ! exe file not
made

```

```

x         goto finish2
x !
x found:
x         write resultFile "( OK (",line,"))"
x         goto finish2
x !
x finish1:
x         close resultFile
x         delete sys#login:linkFile.tmp.*
x         exit
x !
x finish2:
x         close resultFile
x         close file
x         delete sys#login:exeFile.tmp.*
x         delete sys#login:linkFile.tmp.*
x         exit
x !
x error:
x         @[gslws.server]error sys#login:link.res '#STATUS'
x         deassign sys#error

```

RUNJOB.COM

```

x ! runjob.com 8/11/86
x ! this file is used to run an interactive job
x ! job is the name of the user's com file
x ! P2 is the list of appended parameters (optional)
x ! If there is no error in running the job, an OK message is
x ! written out to sys#login:runjob.res.
x ! If there are warnings during the running of the job, an ERROR
message
x ! is returned in sys#login:runjob.res, and the detailed
warning
x ! messages are returned in sys#login:runjob.err
x ! If there is an error in the running of the job, the
abbreviated
x ! error message is returned, from the ERROR.COM file, in
x ! sys#login:runjob.res, and the detailed error message is
written
x ! to sys#login:runjob.err.

x !SET VERIFY
x         job = f#parse("'P1'",,, "name")
x         delete sys#login:runJob.err.*
x         delete sys#login:runJob.res.*
x         define sys#error sys#login:runJob.err
x         on error then goto error

x         @'P1' 'P2'
x         deassign sys#error
x         open/write resultFile sys#login:runJob.res

```

```

* !
* !   If a warning occurs, it is written out to runJob.err
* !   Such warnings are handled specially, through the
* !   runwarning entry.
* !
* SET NOVERIFY
*     define sys$output sys@login:runFile.tmp
*     dir/date/siz sys@login:runJob.err
*     deassign sys$output
* !SET VERIFY
*     open/read file sys@login:runFile.tmp
* !
* loop:
*     read/end_of_file=continue file line
*     show sym line
*     name=f%extract(0,6,line)
*     show sym name
*     if name .eqs. "RUNJOB" then goto runwarning
*     goto loop
* !
* continue:
*     write resultFile "( OK (",job,"      ",P1," ) )"
*     goto finish
* !
* runwarning:
*     message="warning(s) occurred"
*     write resultFile "(ERROR ""'message'" " RUNJOB.ERR)"
* !
* finish:
*     close file
*     close resultFile
*     delete sys@login:runFile.tmp.*
*     exit
* !
* error:
*     @[gslws.server]error sys@login:runJob.res 'STATUS'
*     deassign sys$error

```

STATUS.COM

```

* ! get status of batch jobs
* ! If jobNumber is specified, return only status of that job
* ! If jobNumber is not specified, return all jobs
* ! called by: @status jobNumber
*
*     delete status.res.*
*
*     define sys$output status.tmp
*     show system/batch
*     deassign sys$output
* !SET VERIFY
*

```

```

x      open/read file status.tmp
x      open/write result status.res
x      write result "( OK ("
x      if P1 .eq. "" then goto writeall
x
x loop:
x      read/end_of_file=done file line
x      job = finteger(fextract(15,4,line))
x      if job .eq. P1 then goto found
x      goto loop
x
x done:
x      write result "NIL"          ! no data for specified job
x      goto finish
x
x found:
x      time = fextract(49,11,line)
x      write result "( (JOB ''P1') (CPU ''time') )"
x      goto finish
x
x writeall:
x      read/end_of_file=finish file line
x      jobType = fextract(9,5,line)
x      if jobType .nes. "BATCH" then goto writeall
x      job = finteger(fextract(15,4,line))
x      time = fextract(49,11,line)
x      write result "( (JOB ''job') (CPU ''time') )"
x      goto writeall
x
x finish:
x      write result ") )"
x      close result
x      close file
x      delete status.tmp;
x      exit

```

SUBCOM.COM

```

x ! subcom.com
x ! this is the file actually submitted by submitjob.com
x ! Parameter P1 is the name of the user's COM file to be run
x ! Parameters P2,P3, etc are passed from P3,P4, etc. in
x ! SubmitJob.com
x ! jobname is in the form BATCH_xxx
x ! job is the number (xxx)
x ! if there is an error in the running of the batch job, the
x ! detailed
x ! error message gets sent to 'job'.err.
x ! The abbreviated error message gets sent to 'job'.res

x ! SET VERIFY
x      jobname = fprocess()

```



```

x      job = fextract(6,f=length(jobname)-6,jobname)
x !    open/write outfile junk.
x !    write outfile jobname," ",job
x !    close outfile
x      define syserror 'job'.err
x      on error then goto error

x      @'P1' 'P2' 'P3' 'P4' 'P5' 'P6'
x      exit
x  error:
x      @[gslws.server]error 'job'.res 'STATUS'
-----

```

SUBMITJOB.COM

```

x !    submitjob.com 8/11/86
x !    submit a job on specified queue
x !    call by: @submitjob file queue parameterString
x !          P1 is the file name of the job to be submitted
x !          P2 is the queue (eg., fast, medium)
x !          P3, P4, P5, etc. are subsidiary parameters, such as file
x !          names (eg., file1.dat, file2.sav).
x !    these files are returned in the user's root directory:
x !          P1.log for log file
x !          submitjob.res for result (job # or error message)
x !          submitjob.err for detailed errors (from syserror)
x !          submitjob.tmp for temporary output
x !    these files are returned in the user's running
(sub)directory:
x !          'jobnumber'.res for error message to be returned
x !          'jobnumber'.err for detailed error message

x !SET VERIFY
x      job=fparse("'P1'",,"name")
x      delete syslogin:'job'.log.*
x      delete syslogin:submitjob.err.*
x      delete syslogin:submitjob.res.*
x      delete syslogin:submitjob.tmp.*
x      errorFile = "submitjob.err"
x      tempFile = "submitjob.tmp"
x      resultFile = "submitjob.res"
x      define syserror 'errorFile'
x      on error then goto error

x ! submit the batch job
x SET NOVERIFY
x      if P3.eqs."" then goto zeropar
x      if P4.eqs."" then goto onepar
x      if P5.eqs."" then goto twopar
x      if P6.eqs."" then goto threepar
x      if P7.eqs."" then goto fourpar
x      if P8.eqs."" then goto fivepar
x      goto abort

```

```

x zeropar:
x     define sys%output 'tempFile'
x     submit/noprint/name='job'/parameters=('P1')-
x       /queue='P2' [gslws.server]subcom.com
x     deassign sys%output
x     goto finish

x onepar:
x     define sys%output 'tempFile'
x     submit/noprint/name='job'/parameters=('P1','P3')-
x       /queue='P2' [gslws.server]subcom.com
x     deassign sys%output
x     goto finish

x twopar:
x     define sys%output 'tempFile'
x     submit/noprint/name='job'/parameters=('P1','P3','P4')-
x       /queue='P2' [gslws.server]subcom.com
x     deassign sys%output
x     goto finish

x threepar:
x     define sys%output 'tempFile'
x     submit/noprint/name='job'/parameters=('P1','P3','P4','P5'
x)-
x       /queue='P2' [gslws.server]subcom.com
x     deassign sys%output
x     goto finish

x fourpar:
x     define sys%output 'tempFile'
x     submit/name='job'/parameters=('P1','P3','P4','P5','P6')-
x       /noprint/queue='P2' [gslws.server]subcom.com
x     deassign sys%output
x     goto finish

x fivepar:
x     define sys%output 'tempFile'
x     submit/name='job'/parameters=('P1','P3','P4','P5','P6','P
x7')-
x       /noprint/queue='P2' [gslws.server]subcom.com
x     deassign sys%output

x finish:
x !SET VERIFY
x ! get job number of submitted job from string in submit.tmp
x     open/read infile 'tempFile'
x     read infile line
x ! line now equals " Job xxxx entered on queue ----"
x     startPosition = f%locate("entry",line)+5
x     endPosition = f%locate(")",line)
x     numDigits = endPosition - startPosition

```

```

x      jobNumber = fextract(startPosition,numDigits,line)
x      close infile
x !    delete 'tempFile';*
x      open/write outfile 'resultFile'
x      write outfile "( OK (",jobNumber, " ", P1, " ))"
x      close outfile

x ! no (ERROR ...) message, so deassign the error file
x      deassign syserror
x      exit

x abort:
x      open/write outfile 'errorFile'
x      write outfile "Too many job parameters (more than five)"
x      close outfile
x      deassign syserror
x      exit

x ! get error message
x error:
x      @user1:[gslws.server]error 'resultFile' 'STATUS'
x      deassign sysoutput
x      deassign syserror
x !    delete 'tempFile';*
x      exit
-----

```