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
```

```
¤  error:
¤        @[gslws.server]error sys¤login:abortJob.res '¤STATUS'
¤        deassign sys¤error
¤        close result
¤        close file
¤        delete abortJob.tmp.*
-----------------------------------
```

COMPILE.COM

```
¤ !  COMPILE.COM      8/7/86
¤ ! this file is used to compile a job interactively.
¤ !  job is the name of the user's fortran source file
¤ ! the file exists in the user's local directory, which may be a
¤ !  subdirectory of the root directory.
¤ ! the object file is made in the user's local directory.
¤ ! if there is no error in compilation, the name and date of the
¤ !   object file are returned in sys¤login:compile.res.
¤ ! if there is an error in compilation, the abbreviated error
message
¤ !   is returned in sys¤login:compile.res, and the detailed
¤ !   error message is written to sys¤login:compile.err.
¤
¤
¤ !SET VERIFY
¤        job = f¤parse("''P1'",,,"name")
¤        userDirectory = f¤parse("''P1'",,,"directory")
¤       length=f¤length(job)
¤ !      show symbol job
¤ !      show symbol userDirectory
¤ !     show sym length
¤        delete sys¤login:compile.err.*
¤        delete sys¤login:compile.res.*
¤        delete 'P1'.obj.*
¤        define sys¤error sys¤login:compile.err
¤ !      define sys¤error sys¤login:'job'.err
¤        on error then goto error
¤
¤        fortran/object='userDirectory''job' 'P1'
¤        deassign sys¤error
¤  SET NOVERIFY
¤       define sys¤output sys¤login:objFile.tmp
¤       dir/date 'P1'.obj
¤       deassign sys¤output
¤  !SET VERIFY
¤        open/write resultFile sys¤login:compile.res
¤       open/read file sys¤login:objFile.tmp
¤
¤ loop:
¤       read/end_of_file=done file line
¤ !      show sym line
¤       name=f¤extract(0,length,line)
¤ !      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 continue1
¤ !
¤ simple:
¤       link/exe='userDirectory''job' 'P1'
¤ !
¤ continue1:
¤       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
```

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

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

RUNJOB.COM

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

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

¤        @'P1' 'P2'
¤        deassign sys¤error
¤        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
¤
```

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

SUBCOM.COM

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

¤ !  SET VERIFY
¤        jobname = f¤process()
```

```
¤        job = f¤extract(6,f¤length(jobname)-6,jobname)
¤ !      open/write outfile junk.
¤ !      write outfile jobname,"  ",job
¤ !      close outfile
¤        define sys¤error 'job'.err
¤        on error then goto error

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

SUBMITJOB.COM

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

¤ !SET VERIFY
¤        job=f¤parse("''P1'",,,"name")
¤        delete sys¤login:'job'.log.*
¤        delete sys¤login:submitjob.err.*
¤        delete sys¤login:submitjob.res.*
¤        delete sys¤login:submitjob.tmp.*
¤        errorFile = "submitjob.err"
¤        tempFile = "submitjob.tmp"
¤        resultFile = "submitjob.res"
¤        define sys¤error 'errorFile'
¤        on error then goto error

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

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

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

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

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

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

¤ fivepar:
¤        define sys¤output 'tempFile'
¤        submit/name='job'/parameters=('P1','P3','P4','P5','P6','P
7')-
            /noprint/queue='P2' [gslws.server]subcom.com
¤        deassign sys¤output

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

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

¤ ! no (ERROR ...) message, so deassign the error file
¤         deassign sys¤error
¤         exit


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

¤ ! get error message
¤ error:
¤         @user1:[gslws.server]error 'resultFile' '¤STATUS'
¤         deassign sys¤output
¤         deassign sys¤error
¤ !       delete 'tempFile';*
¤         exit
-----------------------
```