PROGRAMMING SPECIFICATIONS

SCRIPT
PROGRAM FOR SIMULATING MANY USERS

Date: 17-Dec-71
File: SCRIPT.RNO
Edition: 1

This document reflects the software as of version 1.

The information in this document is subject to change without notice and should not be construed as a commitment by Digital Equipment Corporation.

Actual distribution of the software described in this specification will be subject to terms and conditions to be announced at some future date by Digital Equipment Corporation.

TABLE OF CONTENTS

1.0 PURPOSE

The SCRIPT program allows predetermined sequences of characters to be sent over multiple pseudo-teletypes (PTYs) and thereby allow the simulation of a load on the time-sharing system. The results of a run can be watched on-line and statistics of response time, etc. can be recorded on the system disk.

2.0 JOB CAPABILITY

The SCRIPT program has been written as a reentrant program, the code and text to be sent are in the high segment and can be shared by several jobs, each with a 1K low segment. Each job can control up to 14 jobs, so it becomes possible to simulate a large number of PDP-10 time-sharing users.

3.0 CONTROL FEATURES

The SCRIPT is loaded into the high segment from any PDP-10 input device. In addition to the text to be sent, control commands can be included to determine the speed at which the simulated users will operate.

3.1 Timing Parameters

Parameters subject to variation include type-in time, type-out time, and user "think" time. The type-in and type-out time can be specified as a constant or as a rate, in which case the time would depend on the number of characters sent or received. The "think" time is broken up into two parts: 1) the allowed response time, and 2) "free" time. If the computer gives instantaneous response to a command, the total delay is simply the sum of the two quantities. If response is greater than allowed, "think" time is equal to the "free" time. In between, "think" time is the sum of allowed response plus "free" time minus the actual computer response to the command. One additional time factor allowed is the ability to set a maximum response plus type-out delay for a given command. If the command exceeds this limit, then the job is interrupted by sending two control-C characters over the pseudo teletype. This allows a script to include program loops which are interrupted after a specified time interval. The operator can multiply these delay times by an integer factor. In particular, they can be set to 0 to debug a script without the delays present.

3.2 Repetition of Scripts

An additional SCRIPT language feature allows the SCRIPT to specify how many times it will be executed, so that steady state loading conditions can be measured. The operator can override the repeat count.

3.3 Staggering Load Buildup

When multiple jobs are to follow the SCRIPT, the SCRIPT may specify a staggering interval so that load buildup can be gradual.

3.4 Multiple User Numbers

As an optional feature, the SCRIPT program will convert the "#" character into a two digit octal string equal to the PTY unit number. This will allow one script to log many jobs in under different numbers or for multiple jobs to use seperate files. All standard scripts use this feature to log into [4,777#].

4.2 LOGGING OF RESULTS

The SCRIPT program makes an entry in the log file for each line of PTY data sent to the time-sharing system. The time, identity of a particular job, number of characters sent, number of characters received, number of buffers received, and total response time are all recorded. In addition, delays in sending or simulating teletype output due to poor SCRIPT program response are also recorded. This provides some idea of how the response to the controlling job affected system load and response time measurements. Ideally the controlling job would be real-time, i.e. high priority and locked in core. Delays to the script input/output lighten the overall system load, while delays in receiving requests for more input are measured as if the user job had experienced worse response than it actually did.

As a further check on errors in response time measurements due to poor script program response, a "+" character is output when poor SCRIPT program response may have caused the ot wake function in the monitor to fail. When this happens, the SCRIPT program will wake up based on its sleep interval and not on object job response. Normally, the monitor resets the sleep count when a job running on a

pseudo-teletype needs service. The script program tests the PTY flags, and if the PTY doesn't need service sleeps, ideally, this sleep will be terminated immediately by the monitor. However, should the SCRIPT program be rescheduled between testing the PTY and doing the sleep, this can't happen. In this case, the SCRIPT program could sleep for its maximum interval. This could be up to 5 seconds. Unfortunately, there is no way to tell if this happened. The "+" is set whenever a job gets response and the time of day exceeded the time of day the sleep entry would have terminated. This can occur also due to poor response once the sleep interval has terminated.

4.1 Definition of Response Time

The response time measured is the total time from outputting ("typing") a line over the PTY until the program requests the next line or times out to two control-c characters, less time spent simulating TTY output. Thus, if one line to the system produces 50 lines of output, one response time will be recorded, which is the total time the user was waiting for the computer.

4.2 Overlap and Its Effect on Response Time Measurements

There is one possible difficulty: overlap between output and the next input. Normally an output-bound program is waken up when 8 characters of space remain in the monitor buffer. Thus a 0.8 second delay (at 10 characters/sec) would not cause a pause in type-out. Unfortunately, the PTY transfers a line at a time, so this overlapped time is lost. The net effect is to make the measured response times look a little worse than response times an actual TTY user would see. Incidentally, should the program request input while typeout is in progress, all typeout delay will be completed before acting on the input request.

4.3 Different Buffering Modes

If the SCRIPT specified output as a function of time, then the SCRIPT program waits until the delay is up, then data is transferred over the PTY to the SCRIPT program. When the SCRIPT specifies an output rate, the script program must first read the data from the PTY, determine the delay, and then sleep. Specifying one mode or the other may be appropriate to particular programs due to different buffering modes.

4.4 Analysis of Response Time Results

In any event, the user must analyze response time results with a knowledge of these factors, and an understanding of the overlap between user program buffers, monitor buffers, and SCRIPT buffers. Alternately, if psychological system performance is important, the user can use the system manually and see how cleverly the system is concealing its response. The main purpose of the SCRIPT program is to obtain consistent relative response figures to assess the effect of hardware and software changes in the time-sharing system.

5.0 FORMAT OF SCRIPT FILES

5.1 Command Lines

A SCRIPT file consists of SCRIPT lines which end with a <line feed> character. SCRIPT lines are either command lines or text lines. Command lines are used to control typing and operation of the SCRIPT program. Text lines are sent over the pseudo teletype to run the object jobs.

A command line begins with one exclamation point which is followed by a non-exclamation point character.

5.2 Text Lines

A text line begins with no exclamation point, or with two exclamation points. In this case, only the second is sent to the object job. If a SCRIPT line begins with one up-arrow, then the up-arrow is not sent; instead the next character is converted to a control character by complementing bit 100. Two up-arrows result in one up-arrow being sent. (The CR/LF at the end of any line starting with a single up-arrow are not sent.)

Note that "!" and "↑" have no significance except at the beginning of a text or command line.

5.3 Command Line Syntax

Command lines consist of numbers and letter switches. Switches that take numerical values may be preceeded by a number. A number consists of possibly one minus sign followed by a string of decimal digits. There can be no spaces or tabs between the start and end of a number. At other points, spaces and tabs are ignored in command lines.

Should a number not be specified, the last number supplied is assumed. If no value has been supplied, a zero is assumed at the start of each command line.

Certain letter switches may not have numbers associated with them. These switches may not be preceeded by a number unless there is an intervening letter switch that allows a number.

Illegal characters or bad syntax result in an error when processing jobs. These errors are not detected when the SCRIPT is loaded into the high segment.

A command line may include a comment by using a semicolon. A double semicolon and any characters remaining on the command line are deleted when the SCRIPT is loaded and so use no core at run-time. The first 6 characters following the first percent sign of the double semicolon comment of the first SCRIPT line (if any) are taken as the SCRIPT's version number (usually "%nnn").

A command line must not end with a number. Thus a letter switch must follow any number in a command line.

5.4 Letter Switches

i	If non-negative argument supplied	-- sets type-in delay
	If negative argument supplied	-- sets type-in rate.
o	same as I except that it sets type-out rate.	
q	If non-negative argument supplied	-- sets allowed response time
r	If non-negative argument supplied	-- sets free time
s	If non-negative argument supplied	-- sets stagger time
t	If positive argument supplied	-- sets number of times to do SCRIPT.
c	If a positive argument supplied	-- sets maximum delay before program sends ^c^c
	If zero argument supplied	-- inhibits sending ^c^c time-out
I	no arguments allowed	-- sets I mode flag
N	no arguments allowed	-- clears I mode flag
U	no arguments allowed	-- sets U mode flag
V	no arguments allowed	-- clears U mode flag
e	If positive argument supplied	-- sets number of expected errors
	If zero argument supplied	-- inhibits error checking
	(applies for next line only)	
X	no arguments allowed	-- rest of line is comment to operator

5.5 Timing Units

Times are in milliseconds. Rates are in milliseconds per character.

5.6 L Mode

When L mode is set <CR> and <LF> will not be sent over the PTY, thus a line ending with <ALTMODE> can be sent. Any line must end however with a full character set break character due to a limitation in the PTY.

5.7 U Mode

When set U mode converts "#" in text lines into a string of two octal digits equal to the PTY unit number. (Leading zeros will be included.) All DEC SCRIPTs use this feature to log in the job under PPN 4,77700 through 4,77777 (i.e., 4,777#).

5.8 Initial Values

At the start of each pass over a SCRIPT, the following parameters are set up. All remaining quantities are set to zero.

TYPE IN RATE	.3 SEC/CHAR
TYPE OUT RATE	.1 SEC/CHAR
ALLOWED RESPONSE TIME	10 SEC
FREE TIME	7 SEC
STAGGER TIME	20 SEC/JOB
REPEAT COUNT	1 PASS
U MODE ON, L MODE OFF	

5.9 Sample SCRIPT

```
!!;SAMPLE SCRIPT/PFC  25 JULY 1970          IDENTIFY SCRIPT
LOGIN                                       START LOGIN
4,777#                                      USE 4,77720+
XXXX#                                       PASSWORD
!XIN                                        TELL OPERATOR
DEL *.*                                     CLEAR DIRECTORY
MA FOO.F4
        TYPE 1
!       FORMAT (' HI')
        END
!L                                          TERMINATE
```

```
.EXSS
IN
EXECUTE FOO
KJOB/B
!XOUT
```

START LOGOUT
TELL OPERATOR

4.0 OPERATING INSTRUCTIONS

1. Build a time-sharing system with sufficient jobs and pseudo-teletypes. One control job will be needed for each 14 jobs following the SCRIPT.

2. Create an appropriate SCRIPT.

3. Start the SCRIPT program.

4. It will type out the name of the currently loaded SCRIPT. If the segment is sharable, it will go to step 10.

5. If the segment is not sharable, it asks "WHERE TO LOAD A SCRIPT?". Type the device, file name, extension, and directory. If arguments are not supplied, they default to DSK:.SCP

6. The program will respond with "NAME LOADED XX LINES" and return to monitor command level. The name of the job will be changed to the script name. If there were errors it will ask instead if a SCRIPT is to be loaded. Go back to step 5.

7. If the script is not going to be shared, type "CONT" and go to step 10.

8. Save the loaded SCRIPT with a SSAVE command. Example: "SSAVE DSK:FOO<CR>".

9. Start n copies of the program by logging in n jobs and giving the command "RUN DSK FOO" or what ever is needed to load the saved version from step 8. The instructions in the remaining steps should be followed for each of the n jobs.

10. The program will now ask how many jobs are to be run. Enter the number for each control job on its TTY. The maximum number is 14 jobs per control job. Each object job needs a PTY.

11. Should there be too few PTYs, then the SCRIPT program will release all gotten so far and ask over again how many jobs are to be run.

12. The first 1-14 jobs run by a given SCRIPT job can be monitored on any device, this is useful for debugging a new SCRIPT. Answer the question "WATCH HOW MANY JOBS?" appropriately. A null response (just <CR>) will result in all jobs being monitored. If you answer with a number more than 0, it will ask "WHERE". Respond with device, file name, extension, and directory, the defaults are DSK:.WCH. Should the device be unavailable, or the file unenterable, an error message will appear on the teletype and the question will be asked again.

    Should an error occur on the device while running, the run will not be suspended. However, subsequent monitoring will be inhibited. If more than one job is being monitored, each line is preceeded by the process number (01-14, decimal). Except on a teletype monitor, each line will be indented 2*I spaces.

13. If the SCRIPT calls for faster output than the device can handle, then the SCRIPT job will go into I/O wait. This will result in very long delay times attributed to the SCRIPT program. This condition should be avoided by changing the SCRIPT parameters, using a faster device or not monitoring at all.

14. The program will now ask, "WHERE TO LOG RESPONSE TIMES?" Answer with a device, file name, extension, and directory. Arguments not supplied will default to DSK:.RSP. A null response (just <CR>) will result in no responses being logged.

    If the device is not available, or if the file can not be entered, go back to step 14.

    Should a data error occur while running, the run will proceed with subsequent response logging inhibited. An error message will appear on the teletype.

15. SCRIPT asks "REPEAT COUNT = ". Type a decimal number. If zero or null, the repeat count in the SCRIPT will be used.

16. SCRIPT asks "DELAY FACTOR = ". Type a decimal number, this multiplies all the delay times. If 1 or null (just <CR>), the SCRIPT is taken literally, if 0, delays are ignored.

17. SCRIPT asks "STAGGER TIME = ". Type a decimal number. If null, the value in the SCRIPT will be used. If 0, no staggering will apply. If non-zero, the value typed in will be taken (in seconds).

18. SCRIPT asks "SEE COMMENTS TO OPERATOR? ". Type yes or no. If yes, all x SCRIPT commands will be typed as processed prefixed by the process number (1-14) in decimal within parenthesis.

19. If SCRIPT was assembled with the debug switch on (the normal case), it will ask "DEBUG? ", answer yes or no. If you answer yes, then SCRIPT will operate in DEBUG mode, this will add to the monitor file (under monitors with the image mode scanner service) the job status each time it is noticed to be different. The output will be in the form "[\nn\]" where nn is the high order six bits returned by the JOBSTS UUO. These bits are as follows:

40	job number assigned
20	job completed login, not started logout
10	TTY at monitor level
04	TTY output available
02	TTY in need of input to do anything
01	control c defeated

20. SCRIPT asks "WATCH FOR ERRORS?", Type yes or no. If yes, all PTY output for all the jobs will be monitored for "?" lines. Unless the SCRIPT has predicted errors, all such lines will be typed on the teletype prefixed by the process number (1-14) in decimal within parentheses and followed by two bells.

21. SCRIPT will say "STARTING". Jobs under control of a given control program will be staggered by the interval specified by the operator or in the SCRIPT. If multiple SCRIPT jobs are running, the user can stagger their timing by ending step 20 at the appropriate time of day.

22. When the last job under a given SCRIPT program has finished, the message "ALL JOBS DONE" will appear on the users teletype and the log file and monitor file will be closed. The program will then exit.

23. Should it be necessary to terminate a run, stop the control program by typing two <control c> characters. To close out the log file reenter the program by a reenter command. A message "JOBS ABORTED" will appear on the teletype, and all jobs will be kJobbed.

24. The FORTRAN program TOTAL can be used to summarize the response log file. It takes input from 1:F3R01.DAT and places a short listing on 6:F3R06.DAT.

4.1 Sample Dialogue

```
R SCRIPT                                    GET SCRIPT
NO SCRIPT LOADED
WHERE TO LOAD A SCRIPT? SYS:DELALL          NAME OF SCRIPT
DELALL.SCP%002 LOADED 15 LINES

.SS DSK:DELALL                              SAVE SO CAN SHARE
JOB SAVED
*C

.RU DSK:DELALL                              GET SHARABLE COPY
SCRIPT DELALL.SCP%002 LOADED
HOW MANY JOBS TO BE RUN? 14                 MAX NUMBER
WATCH HOW MANY JOBS?                        DO ALL
WHERE? DEL1                                 USE DSK:DEL1.WCH
WHERE TO LOG RESPONSE TIMES? DEL1           USE DSK:DEL1.RSP
REPEAT COUNT =                              USE SCRIPT VALUE
DELAY FACTOR =                              USE SCRIPT VALUE
STAGGER TIME =                              USE SCRIPT VALUE
SEE COMMENTS TO OPERATOR? Y                 WATCH PROGRESS
DEBUG? N                                    NOT USUALLY NEEDED
WATCH FOR ERRORS? Y                         A GOOD IDEA
STARTING
(1) IN                                      JOB 1 LOGGED IN
(2) IN                                      JOB 2 LOGGED IN
...
(13) OUT                                    JOB 13 LOGGED OUT
(14) OUT                                    JOB 14 LOGGED OUT
ALL JOBS DONE                               IT WAS FINISHED
EXIT
*C
```