UNIX Remote Job Entry User's Guide

A. L. Sabsevitz
K. A. Kelleman

Bell Laboratories
Piscataway, New Jersey 08854

1. PREFACE

A set of background processes running under UNIX† support remote job entry to IBM System/360 and /370 host computers. RJE is the communal name for this subsystem.[1] UNIX communicates with IBM's Job Entry Subsystem by mimicking an IBM 360 remote multileaving work station. The *UNIX User's Manual* entry *rje*(8) summarizes its design and operation. The manual also contains a description of the *send*(1C) command, which is the user's primary method of submitting jobs to RJE, and *rjestat*(1C), which allows the user to monitor the status of RJE and to send operator commands to the host system. This guide is a tutorial overview of RJE and is addressed to the user who needs to know how to use the system, but does *not* need to know details of its implementation. The two following sections constitute an introduction to RJE.

2. PRELIMINARIES

To become a UNIX user, you must receive a login name that identifies you to the UNIX system. You should also get a copy of the *UNIX User's Manual;* it contains a fairly complete description of the system and includes the section *How to Get Started*, which introduces you to UNIX; you should read that section before proceeding with this guide.

In order to begin using RJE, you need only become familiar with a subset of basic commands. You must understand the directory structure of the file system, and you should know something about the attributes of files: see *cd*(1), *chmod*(1), *chown*(1), *cp*(1), *ln*(1), *ls*(1), *mkdir*(1), *mv*(1), *rm*(1). You must know how to enter, edit, and examine text files: see *cat*(1), *ed*(1), *pr*(1). You should know how to communicate with other users and with the system: see *mail*(1), *mesg*(1), *who*(1), *write*(1). And, finally, you might have to know how to describe your terminal to the system: see *ascii*(5), *stty*(1), *tabs*(1).

3. BASIC RJE

Let's suppose that you have used the editor, *ed*(1), to create the file, **jobfile**, that contains your job control statements (JCL) and input data. This file should look exactly like a card deck, except that for convenience alphabetic characters may be in either upper or lower case. Here is an example:

---

† UNIX is a trademark of Bell Laboratories.

1. In this paper, RJE refers to the facilities provided by UNIX, and *not* to the Remote Job Entry feature of IBM's HASP and JES subsystems.

```
$ cat jobfile
//gener job (9999,r740),pgmrname,class=x usr=(mylogin,myplace)
//step exec pgm=iebgener
//sysprint dd sysout=a
//sysin dd dummy
//sysut2 dd sysout=a
//sysut1 dd *
   first card of data
        :
        :
   last card of data
/*
```

To submit this job for execution, you must invoke the *send*(1C) command:

```
$ send jobfile
```

The system will reply:

```
10 cards
Queued as /usr/rje/rd3125
```

Note that *send* tells you the number of cards it submitted and reports the file name that contains your job in the queue of all jobs waiting to be transmitted to the host system. Until the transmission of the job actually begins, you can prevent the job from being transmitted by doing a **chmod 0** on the queued file to make it unreadable. For our example, you could say:

```
chmod 0 /usr/rje/rd3125
```

When your job is accepted by the host system, a job number will be assigned to it, and an acknowledgement message will be generated. This indicates that your job has been scheduled on the host system. Later, after the job has executed, its output will be returned to the UNIX system. You will be notified automatically of both of these events: if you are logged in when RJE detects these events, and if you are permitting messages to be sent to your terminal (see *mesg*(1)). The following two messages will be sent to you (still using the example above) when the job is scheduled and when the output is returned, respectively:

```
    Two bells
12:18:42 gener job 384 -- rd3125 acknowledged

    Two bells
12:21:54 gener job 384 -- /a1/user/rje/prnt0 ready
```

Two bells, with an interval of one second between them, precede each message. They should be interpreted as a warning to stop typing on your terminal, so that the imminent message is not interspersed with your typing.

If you are not logged in when one of these events occurs, or if you do not allow messages to be sent to your terminal, then the notification will be posted to you via the *mail*(1) command. You can prevent messages directly by executing the *mesg*(1) command, or indirectly by executing another command, such as *pr*(1), which prohibits messages for as long as it is active. You may inspect (by invoking the *mail* command) your mail file (/usr/mail/*logname*) at any time for messages that have been diverted. Setting your **MAIL** variable to the name of your mail file will cause the shell to notify you when mail arrives. For this example, the mail might look as follows:

```
$ mail
From rje Mon Aug  1 12:20:36 1977
12:18:42 gener job 384 -- rd3125 acknowledged

? d
From rje Mon Aug  1 12:21:55 1977
12:21:54 gener job 384 -- /a1/user/rje/prnt0 ready

? d
```

The job acknowledgement message performs two functions. First, it confirms the fact that your job has been scheduled for eventual execution. Second, it assigns a number to the job in such a way that the number and the name together will uniquely identify the job for some period of time.

The output ready message provides the name of a UNIX file into which output has been written and identifies the job to which the output belongs (see *ls*(1)):

```
$ ls -l prnt0
-r--r-xr--  1 rje          1184 Aug  1 12:21 prnt0
```

Note that rje retains ownership of the output and allows you only read access to it. It is intended that you will inspect the file, perhaps extract some information from it, and then promptly delete it (see *rm*(1)):

```
$ rm -f prnt0
```

The retention of machine-generated files, such as RJE output, is discouraged. It is your responsibility to remove files from your RJE directory. RJE output files may be truncated if the output exceeds a set limit. This limit is tunable by the system administrator. Output beyond the current limit will be discarded, with no provision for retrieval. If the output were truncated in the previous example, the second notification message would have been:

```
        Two bells
12:21:54 gener job 384 -- /a1/user/rje/prnt0 ready (truncated)
```

The user should also be aware that RJE attempts to keep a set number of blocks free on any file system it uses. This number is also tunable by the system administrator. Warning messages or suspension of certain functions will occur as this limit is approached.

The most elementary way to examine your output is to *cat* it to your terminal. The Appendix of this document shows the result of listing the output of our sample job in this way. Because UNIX has no high volume printing capability, you should route to the host's printer any large listings of which you desire a hard copy.

The structure of an output listing will generally conform to the following sequence:

```
HASP log
jcl information
data sets
HASP end
```

Normally burst pages will not be present. Single, double, and triple spacing is reflected in the output file, but other forms controls, such as the skip to the top of a new page, are suppressed. Page boundaries are indicated by the presence of a blank (space character) at the end of the last line of each page.

The big file scanner *bfs*(1) or the context editor *ed*(1) provide a more flexible method than *cat*(1) for examining printed output; *bfs* can handle files of any size and is more efficient than *ed* for scanning files.

RJE is also capable of receiving punched output as formatted files (see *pnch*(5)); this format allows an exact representation of an arbitrary card deck to be stored on the UNIX machine. However, there are few commands that can be used to manipulate these files. You will probably want to route your punched output to one of the host's output devices.

4. SEND COMMAND

The *send*(1C) command is capable of more general processing than has been indicated in the previous section. In the first place, it will concatenate a sequence of files to create a single job stream. This allows files of JCL and files of data to be maintained separately on the UNIX machine. In addition, it recognizes any line of an input file that begins with the character ~ as being a *control* line that can call for the inclusion, inside the current file, of some other file. This allows you to *send* a top level skeleton that "pulls" in subordinate files as needed. Some of these may be "virtual" files that actually consist of the output of UNIX commands or Shell procedures. Furthermore, the *send* command is able to collect input directly from a terminal, and can be instructed to prompt for required information.

Each source of input can contain a format specification that determines such things as how to expand tabs and how long can an input line be. The manual entry for *fspec*(5) explains how to define such formats. When properly instructed, *send* will also replace arbitrarily defined keywords by other text strings or by EBCDIC character codes. (These two substitution facilities are useful in other applications besides RJE; for that reason, *send* may be invoked under the name *gath* to produce standard output *without* submitting an RJE job.)

Two options of *send* that everyone should be acquainted with are: the ability to specify to which host computer the job is to be submitted, and a flag that guarantees that a job will be transmitted to the host computer in order of submission (relative to other jobs submitted with the same flag). To run our sample job on a host machine known to RJE as **A**, we would issue the command:

```
$ send A jobfile
```

When no host is explicitly cited, *send* makes a reasonable choice.

To insure that a job will be transmitted in order of submission, set the **−x** flag:

```
$ send −x jobfile
```

This flag should be used sparingly. The complete list of arguments and flags that control the execution of *send* can be found in *send*(1C).

5. JOB STREAM

It is assumed that the job stream submitted as the result of a single execution of *send* consists of a single *job*, i.e., the file that is queued for transmission should contain one JOB card near the beginning and no others. A priority control card may legitimately precede the JOB card. The JOB card must conform to the local installation's standard. At BISP, it has the following structure:

```
//name job (acct[,...]),pgmrname[,keywds=?] [usr=...]
```

6. USER SPECIFICATION

A "usr" specification is required on print or punch output that is to be delivered to a UNIX user.

```
usr=(login,place,[level])
```

where *login* is the UNIX login name of the user, *level* is the desired level of notification (see end of this section for an explanation), and *place* is as follows:

A. If *place* is the name of a directory (writable by others), then the output file is placed there as a unique **prnt** or **pnch** file. The mode of the file will be 454.

B. If *place* is the name of an existing, writable (by others), non-executable (by others) file, then the output file replaces it. The mode of the file will be 454.

C. If *place* is the name of a non-existent file in a writable (by others) directory, then the output file is placed there. The mode of the file will be 454.

D. If *place* is the name of an executable (by others) file, then the RJE output is set up as standard input to *place*, and *place* is executed. Five string arguments are passed to *place*. For example, if *place* is a shell procedure, the following arguments are passed as $1 ... $5:

1. Flag indicating whether file space is scarce in the file system where *place* resides. A **0** indicates that space is *not* scarce, while **1** indicates that it is.
2. Job name.
3. Programmer's name.
4. Job number.
5. Login name from the "usr=..." specification.

A ":" is passed if a value is not present. The current directory for the execution of *place* will be set to the directory containing *place*. The environment (see *environ*(7)) will contain values for **LOGNAME** and **HOME** based on the login name from the "usr=..." specification, and a value for **TZ**. Since the login name supplied on the "usr=..." specification cannot be believed for security purposes, the UID will be set to a reserved value.

E. In all other cases, the output will be thrown away.

The *place* value must not be a full path name, unless it refers to an executable file (see D above). For cases A, B, and C above (and case D, if a full path name is not supplied), the name of the user's login directory will be used to form a full path name.

The "usr=..." field may occur anywhere within the first 100 card images sent and within the first 200 output images received by the UNIX system. The only restriction is that it be contained completely on a single line or card image. Therefore, the "usr=..." field may be placed on a JOB card or comment card. It may also be passed as data.

For redirection of output by the host, a "usr=..." card, if not already present, must be supplied by the user. This can be done by placing a job step that creates this card before your output steps.

Messages generated by RJE or passed on from the host are assigned a level of importance ranging from 1 to 9. The levels currently in use are:

3 transmittal assurance
5 job acknowledgement
6 output ready message

The optional *level* field of the "usr=..." specification must be a one or two-digit code of the form *mw*. A message from the host with importance *x* (where *x* comes from the above list) is compared with each of the two decimal digits in *level*. If $x \geq w$ and if the user is logged in and is accepting messages, the message will be written to his or her terminal. Otherwise, if $x \geq m$, the message will be mailed to the user. In all other cases, the message will be discarded. The default *level* is **54**. You should specify level **1** if you want to receive complete notification, and level **59** to divert the last three messages in the above list to your mailbox.

7. MONITORING RJE

RJE is designed to be an autonomous facility that does not require manual supervision. RJE is initiated automatically by the UNIX reboot procedures and continues in execution until the system is shut down. Experience has shown RJE to be reasonably robust, although it is vulnerable to system crashes and reconfigurations.

Users have a right to assume that when the UNIX system is up for production use, RJE will also be up. This implies more than an ability to execute the *send*(1C) command, which should be available at all times; it means that queued jobs should be submitted to the host for execution and their output returned to the UNIX system. If a user cannot obtain any throughput from RJE, he or she should so advise the UNIX operators.

The *rjestat*(1C) command, invoked with no arguments will report the status of all RJE links for which a given UNIX system is configured. It may sometimes also print a message of the day from RJE.

```
$ rjestat
RJE to B operating normally.
RJE to A down, reason: IBM not responding.
```

A host machine may be reported to be not responding to RJE because it is down, or because of its operator's failure to initialize the associated line, or because of a communications hardware failure.

Rjestat also has the ability to send operator commands to the host machine and retrieve the responses generated by the commands. Refer to the *rjestat*(1C) manual entry for a complete description of this command.

APPENDIX

Sample JES2 Output Listing

```
$ cat rje/prnt0
14.40.31 JOB 384  $HASP373 GENER  STARTED - INIT 26 - CLASS X - SYS RRMA
14.40.32 JOB 384  $HASP395 GENER  ENDED

------ JES2 JOB STATISTICS ------

 1 AUG 77 JOB EXECUTION DATE

          54 CARDS READ

          76 SYSOUT PRINT RECORDS

           0 SYSOUT PUNCH RECORDS

       0.01 MINUTES EXECUTION TIME
    1        //GENER JOB (9999,R740),PGMRNAME,CLASS=X                    JOB 384
             ***       USR=(MYLOGIN,MYPLACE)
    2        //IEBGENER EXEC PGM=IEBGENER
    3        //SYSPRINT DD DUMMY
    4        //SYSIN DD DUMMY
    5        //SYSUT2 DD SYSOUT=A
    6        //SYSUT1 DD *
             //
IEF236I ALLOC. FOR GENER IEBGENER
IEF237I DMY ALLOCATED TO SYSPRINT
IEF237I DMY ALLOCATED TO SYSIN
IEF237I JES ALLOCATED TO SYSUT2
IEF237I JES ALLOCATED TO SYSUT1
IEF142I GENER IEBGENER - STEP WAS EXECUTED - COND CODE 0000
IEF285I   JES2.JOB0384.SO0102                          SYSOUT
IEF285I   JES2.JOB0384.SI0101                          SYSIN
IEF373I STEP /IEBGENER/ START 77242.1440
IEF374I STEP /IEBGENER/ STOP 77242.1440 CPU 0MIN 00.13SEC SRB 0MIN 00.01SEC VIRT 36K SYS 188K

******* SERVICE UNITS=0000174          SERVICE RATE=0000268 SERVICE UNITS/SECOND
******* PERFORMANCE GROUP=005
******* EXCP COUNT BY UNIT ADDRESS
IEF375I  JOB /GENER / START 77242.1440
IEF376I  JOB /GENER / STOP  77242.1440 CPU    0MIN 00.13SEC SRB    0MIN 00.01SEC

******* SERVICE UNITS=0000174          SERVICE RATE=0000268 SERVICE UNITS/SECOND
******* APPROXIMATE PROCESSING TIME=            .01 MINUTES
******* EXCPS=000000000
******* PROJECTED CHARGES=              .01

          first line of data
            :
            :
          last line of data

*OS/VS2 REL 3.7 JES2*  END  JOBNAME=GENER   BIN=R740      JOB #=384  PGMRNAME
*OS/VS2 REL 3.7 JES2*  END  JOBNAME=GENER   BIN=R740      JOB #=384  PGMRNAME
*OS/VS2 REL 3.7 JES2*  END  JOBNAME=GENER   BIN=R740      JOB #=384  PGMRNAME

$ rm -f rje/prnt0
```

January 1981