

## Running SPSS Jobs on UNIX

03/27/00

Last Revised:

This document contains the class notes from classes taught to transitioning VMSers. It provides the following instructions:

- SPSS Programs Written for VMS
- Where to Run SPSS on UNIX
- UNIX Commands and the SPSS Command
- Running Jobs in the Foreground and Background
- Managing SPSS Jobs
- Preparing to Run SPSS Jobs.

## SPSS Programs Written for VMS

Very little needs to be changed in your SPSS programs that were written to run on VMS:

1. Need to remove the DCL at the beginning of the file (\$SPSS).
2. Need to change the path names used to read and write files.
3. For reading in ASCII data with a record length greater than 1024 columns, you must use a FILE HANDLE command with the LRECL subcommand which specifies the record width:

```
file handle school/name='school90.dat' /lrecl=1124
```

Warning: The above refers to your SPSS *programs*, not your SPSS system files. You must make SPSS export files **ON VMS** before these files will be readable on UNIX.

## Where to Run SPSS on UNIX

SPSS is only licensed to run on NORMAN. NORMAN is our newest UNIX computer and very powerful, four times faster than GUY.

## UNIX Commands

Every UNIX command has a standard way of getting its input, writing its output, and writing error messages:

- standard input
- standard output
- standard error

Standard input is the place from which commands get their data. By default, this is the keyboard. Standard output is the place that commands put their output. By default, this is the screen. Standard error is the place that commands put their error messages. By default, this is the screen, also. But it is important to note that standard output and standard error are not the same thing. It just happens that, by default, they send data to the same place. Collectively, these are called standard input and output, or standard I/O, abbreviated stdio.

<b>Stdio Elements</b>	<b>Default</b>	<b>Abbreviation</b>
standard input	keyboard	stdin
standard output	screen	stdout
standard error	screen	stderr

Standard I/O can be redirected so that it comes from, or goes to, any place. Standard input can come from the keyboard, or from a file, or from another command. Standard output can be sent to the screen, or to a file, or into another command (as standard input to that command). This is the power of the standard I/O system.

The symbols used to redirect output are:

- >        redirect stdout from screen to a file
- >&        redirect stdout and stderr from screen to a file
- <        redirect stdin from screen to a file

## **Redirection of Standard Output**

One of the most common ways to manipulate standard I/O is to redirect standard output from a command into a file. For example, if you want to save a long listing of one of your directories, you can do this:

```
ls -l > doc.list
```

-l is an option specifying a detailed or long listing. Options begin with a dash (rather than a / like on VMS). You need a space between the command name and the dash.

Equivalent VMS command:

```
dir/full/output=doc.lis
```

The "greater than" sign (>) redirects data from the `ls` command to a file called `doc.list`. Without the redirection, the listing would appear on the screen, but with the redirection, the command only returns the prompt, with no listing. If the file `doc.list` already exists,

then it will be overwritten by the data from the `ls` command.

## SPSS Command

With no options or redirection:

```
spss
```

invokes SPSS in interactive mode.

```
spss -m < myspsscommands.sps > spssoutput.out
```

invokes SPSS in noninteractive mode.

One more little wrinkle:

```
spss -m < myspsscommands.sps >& spssoutput.out
```

This command redirects SPSS error messages into the same file that holds the output. Otherwise, SPSS will print some error messages to the screen only.

## Increasing Memory

SPSS allocates 512K of memory to its data areas by default. You can request additional memory by adding the `-s` option to the `spss` command line. For example, to request two megabytes of memory for the example above, you would type the following:

```
spss -m -s2m <foo.sps >foo.spsout &
```

## Running Jobs in the Foreground and Background

UNIX was designed as a multiprocessing operating system. This means that the computer can perform (or, more accurately, can seem to perform) more than one task at once. While your statistical job is running, you have the option of either waiting while it runs (running the job in the foreground), or doing other tasks while it runs (running the job in the background).

Ordinarily, jobs on UNIX run in the foreground; that is, while the command is running, the prompt is not returned and you must wait for completion of the command before any more work can be performed. However, you may instruct the computer to run the job in the background. When you do this, your prompt is immediately returned and you may do other tasks, or even log off, while the background job is running.

Background processing on UNIX is very similar to batch processing on VMS with a few very important differences:

- Background processes can make the demands on system resources as interactive processes.
- Background processes are not restricted by the same limitations imposed on a VMS batch queue such as CPU time, queue names, etc.

To run a job in the background, place an ampersand at the end of the command:

```
spss -m < little.sps >& little.out &
```

Your prompt will be returned to you immediately and you may do other work (including logging off). When the job successfully completes, you will receive the message:

```
[1] Done spss -m < little.sps >& little.out
```

If the job completes unsuccessfully, you will get a message like:

```
[1] Exit 1 spss -m < little.sps >& little.out
```

The number following the word `Exit` may vary.

It is wise to run no more than one or two jobs in the background at a time, as more jobs will tend to slow down the processing of all jobs.

## Exercise

1. Create a directory called `Spssclass`:

```
mkdir Spssclass
```

2. Change to the `Spssclass` subdirectory:

```
cd Spssclass
```

3. Copy `little.sps` from `mcdermot`'s directory to your `Spssclass` directory:

```
cp ~mcdermot/spssjobs/little.sps .
```

4. Examine the command file:

```
more little.sps
```

5. Run the job:

```
spss -m < little.sps >& little.out &
```

6. Examine the output:

```
more little.out
```

## Moving Jobs from the Foreground to the Background

It's easy to forget to start up jobs in the background. Fortunately though, jobs running in the foreground can be placed in the background, but first, they must be suspended. A job that is suspended is not actually running. It is frozen. It can start up again. When the job restarts, it does not start from the beginning, but rather, it continues running from the point at which it was suspended.

To suspend a job running in the foreground, use the <Ctrl-Z> keystroke. For example:

```
> spss -m < big.sps >& big.out
^Z
Suspended (signal)
```

In this example, SPSS was running in the foreground. The suspend keystroke, <Ctrl-Z>, suspended the job and the prompt was returned. At this point, you can put the job in the background, by executing the `bg` command. For example:

```
> bg
[1] spss -m < big.sps >& big.out &
```

## Managing SPSS Jobs

Occasionally, you may have a few jobs running at once and you may want to list the jobs to see what their status is. To get a listing of jobs, use the `ps` command.

```
> ps -u mcdermot
  PID TTY          S       TIME COMMAND
 4834 ttyp9    I         0:00.04 spsscmd
 6982 ttyp9    T         0:00.07 tpu_v41 big.sps
 7496 ttyp9    S         0:00.74 emacs little.sps
 9572 ttyp9    R         3:06.91 SPSS
10752 ttyp9    I         0:00.07 ksh /usr/local/bin/spss -m
12617 ttyp9    S         0:02.73 -tcsh (tcsh)
```

The `-u` option says that you want a listing of jobs that user `mcdermot` is running. You can put any name in here, but, typically, you will use your own login name. The process identification number for the job is under the `PID` column. This number may be useful, as we will see. The `TIME` column indicated how much CPU time has been used by this process in minutes and seconds. Note that SPSS starts up three processes for each SPSS command specified.

## Killing jobs

Sometimes it is necessary to kill jobs that are running. If the job is running in the

foreground, use the keystroke <Ctrl-C>:

```
> spss -m < big.sps >& big.out  
<Ctrl-C>
```

In this example, an SPSS job is started. Before it is completed, the <Ctrl-C> keystroke is given; the job is killed and the prompt is returned to the screen.

If the job is running in the background or if the job is suspended, use the `kill` command. The `kill` command takes as a parameter the process identification number (PID) as reported by the `ps` command:

```
> ps -u mcdermot  
  PID TTY          S       TIME COMMAND  
21013 ttyq5      R        0:00.45 SPSS  
22094 ttyq5      S        0:02.83 -tcsh (tcsh)  
22689 ttyq5      S        0:00.10 ksh /usr/local/bin/spss -m  
23769 ttyq5      I        0:00.07 spsscmd  
> kill 21013
```

Note, for very technical reasons which we will not want to go into, be sure you kill the process called "SPSS". If you kill one of the other two SPSS processes, your job will not get killed.

If you kill a statistical process, you should check the `/tmp` directory to confirm that all of your temporary files were deleted. To do this, run the command:

```
ls -l /tmp
```

If you own any files in this directory, remove them using the `rm` command. If you own any directories in this directory, remove them using the `rm -r` command.

## Preparing to Run SPSS Jobs

Before you run your SPSS job, there are a few things to consider:

- How busy is NORMAN at the time you want to start an SPSS job?
- Is there enough `/tmp` space to run your SPSS job?
- Where should you write your output?

## Checking the System Load

Use the `uptime` command to check to see how busy the machine is before you run your job. Along with other information, `uptime` displays a machine's load average:

```
> uptime  
9:05am up 19:14, 11 users, load average: 2.36, 2.24,
```

## 2.01

The numbers of interest are the three numbers displayed after load average (2.36, 2.24, and 2.01 in the example above). These numbers represent the load average for the last 5, 30, and 60 seconds. If these numbers are over three (especially the first number), consider running your job at a later time or on another machine because the machine is overloaded already.

Another useful command with even more information is `top`:

```
load averages:  3.14,  2.16,  1.88
219 processes:  5 running, 84 sleeping, 126 idle, 3 stopped
Cpu states: 71.9% user,  0.0% nice, 28.0% system,  0.0% idle
Memory: Real: 201M/304M act/tot  Virtual: 161M/790M use/tot  Free:
5416K
```

PID	USERNAME	PRI	NICE	SIZE	RES	STATE	TIME	CPU	COMMAND
15880	aminkin	49	0	23M	2310K	run	0:00	21.80%	netscape
19287	lu	50	0	44M	5734K	run	33:51	19.50%	<stata>
13395	aminkin	50	0	24M	3940K	run	0:01	18.90%	netscape
5629	jyocom	49	0	16M	9625K	run	0:23	17.10%	netscape
29	root	46	0	184K	24K	sleep	11:59	6.40%	update
17008	aminkin	42	0	1928K	409K	sleep	0:00	3.60%	tcsh
20143	mcdermot	47	0	2848K	524K	run	0:01	2.20%	top
320	root	44	0	1552K	270K	sleep	6:22	0.90%	automount
14130	root	44	0	1448K	294K	sleep	0:07	0.50%	telnetd
31168	epudberr	44	0	4296K	1032K	sleep	0:01	0.50%	pine
283	root	45	0	1456K	155K	sleep	19:01	0.30%	ypserv
28846	akourtel	44	0	1896K	311K	sleep	0:03	0.30%	pico
27312	root	44	0	1448K	294K	sleep	0:02	0.30%	telnetd
235	root	44	0	4232K	1957K	sleep	5:50	0.10%	named
739	root	42	-2	6000K	540K	sleep	1:29	0.10%	Xdec

Use `<Cntl>C` to terminate `top`.

To check the system load on other UNIX computers, use the `rup` command:

```
cdeinfo.ssc.  up 99 days,  1:40,  load average: 0.07, 0.00, 0.00
wingra.ssc.w  up  1 day,   4:29,  load average: 0.11, 0.07, 0.05
elaine.ssc.w  up 99 days,  4:06,  load average: 1.99, 1.33, 1.30
norman.ssc.w  up  1 day,   4:17,  load average: 1.73, 1.98, 1.94
duncan.ssc.w  up  8 days,  4:11,  load average: 2.80, 2.61, 2.22
```

You may need to use `<Cntl>C` to terminate `rup`.

## Checking /tmp Before Running Statistical Software

Unlike most directories on UNIX, each UNIX computer has its own /tmp directory. That is, /tmp on NORMAN is completely different from /tmp on ELAINE. /tmp on each node also may have different capacities.

It is always a good idea to check and see how full the /tmp directory is where you are logged on before you run your SPSS program. To do this you use the `df` command which displays the amount of available space on the specified disk:

```
> df -k /tmp
Filesystem      1024-blocks  Used      Avail      Capacity
/dev/rz2c      1991325      19      1792173      0%
```

The column headed 1024-blocks is the amount of space occupied by the disk in kilobytes, Used is the amount of used space, Avail is the amount of available space, and Capacity is the percentage of the disk's total capacity that has been used.

If the `df` command indicates that /tmp is full or nearly so, you may want to consider running your program later or on another UNIX computer which may have more /tmp space available.

If you would like to check how much space is available in /tmp on one of the UNIX computers that you are not currently logged on to, you can use the `rsh` command. `rsh` connects to a specified computer and executes a specified command. For example:

```
rsh guy df -k /tmp
```

will do a remote log in to GUY and check its /tmp space with the `df` command.

## Where to Write Your Output

You have write access to three different file systems on UNIX:

/home	Home directories 24 GB Backed up 100 MB quota granted upon receiving a new account Quota not expandable
/aux	Additional disk space 12 GB Backed up 200 MB quota granted on request Quota expanded with permission of group leader Quota expandable for 1, 3, or 6 months

Quota expandable to 300, 400, or 500 MB

/temp/fivedays Additional short term disk space for large amounts of data  
4 GB  
NOT BACKED UP  
No quotas: users may use as much space as is available  
Every night, files five days old are removed

Recall that 1mb = 2,000 VMS blocks.

To check your quota, either log on to GUY and type `quota`, or type the following command from any UNIX computer:

```
> rsh guy quota
Disk quotas for user mcdermot (uid 565):
  Filesystem  blocks  quota  limit  grace  files
    /aux      84359  204800 204800         13
    /home/m   80479  102400 102400        6227
```

mcdermot is using 80mb of the 100mb allocated on /home and 84mb of the 200mb allocated on /aux.

To find out how much disk space you are using by directory, type

```
du -k /home/m/mcdermot          or      du -k /aux/m/mcdermot
```

If you process a lot of SPSS jobs, you will want to write to your /aux space:

```
spss -m < big.sps >& /aux/m/mcdermot/dissert/big.out
```

## Scheduling SPSS Jobs

Very large SPSS jobs should be run at off peak hours when few users are using the computer interactively. To run jobs at off peak hours, use the `at` command. The `at` command allows a user to schedule the execution of a job or a series of jobs for a later time. The syntax for the `at` command is as follows:

```
at time
```

The `at` command reads commands from standard input to be executed at the time specified. The user's environment (current directory, environment variables, etc.) are recreated at the later time of execution.

The `at` command is very flexible on the format of the time parameter. Any of the following formats can be used:

```
at 3:52
at 3am
```

```
at 3pm
at noon
at midnight
at 8am Friday
```

The `at` command runs the command at the next time that matches the time parameter. For instance, the command:

```
at 3:52
```

will be run the next time that the time is 3:52. If you submitted this in the morning, it would run at 3:52 in the afternoon. If you submitted this after dinner, it would run at 3:52 the next morning. To be sure to have it run at the desired time, you should specify `am` or `pm`.

After the `at` command has been submitted, the computer will wait for you to enter as many commands as you desire until you give the end-of-file keystroke, the `<Ctrl-D>`. Your prompt will only be returned after you have entered `<Ctrl-D>`. For example:

```
> at 3:52am
spss -m < big1.sps >& big1.out
spss -m < big2.sps >& big2.out
^D
job rrodrigu.873795120.a at Tue Sep 09 03:52:00 1997
```

In this example, two SPSS jobs were submitted. They will run at 3:52 the next morning. The last line before the prompt tells you when the job will run and it gives you a job ID. In this case, the job ID is:

```
rrodrigu.873795120.a
```

The jobs automatically run in the background and so the ampersand is not needed, and, in fact, will be detrimental, as all SPSS jobs will run simultaneously, instead of running sequentially and so slow down the system in general.

The `at` command can also be used to list or remove the jobs that you have submitted. The `-l` option lists jobs that have been submitted by job number:

```
> at -l
rrodrigu.873795120.a    Tue Sep 09 03:52:00 1997
```

The `-r` option removes a job. Give the `at` command the `-r` option and the job number (either from the `at` command itself, or from the output of the `at -l` command). For example:

```
> at -r rrodrigu.873795120.a
at file: rrodrigu.873795120.a deleted
```

## Running Jobs "nice"ly

The `nice` command executes commands at a lower priority, allowing other jobs to get more computer time. It may seem counterintuitive to run a job with a lower priority, but there are several reasons why you might want to do this.

1. You are going away to stop working for the day after you submit the job. Whether the job completes very quickly, or only somewhat quickly is a matter of indifference.
2. Users are working interactively (reading email, running a web browser, for example) and you do not want to slow down their work.
3. You are submitting a very large job expected to run for some time.

In any of these cases, you should run the job at a lower priority, using the `nice` command:

```
nice spss -m < big.sps >& big.out &
```