Mark-5 Auxiliary Programs

Several sometimes-useful auxiliary programs accompany the Mark-5 system. Following are brief descriptions of these. Two programs from Conduant:

`ssopen`

which opens and does a quick test of the SS card to verify readability, and

`sstest`

which *erases* and *writes* a scan of 33554432 bytes of SS pattern onto the SS disks. *Important*: Do *not* use sstest on disks that have data to be saved.

The following programs are Mark-5 specific.

`tstMark5A [ machine ]`

where *machine* defaults to localhost. Mark5A should already be running on *machine*, but `tstMark5A` can run on some other machine including most Linux and HP-UX machines and probably others. This program accepts commands and queries as defined in the Mark5A software documentation, sends them to Mark5A on *machine*, and prints the responses.

`SSErase [ -m m ] [ -c c ] [ -h ]`

where *m* (msglev) is the debug message level, range -1 to 3, default 1, *c* sets conditioning (0 for FALSE, 1 TRUE, default FALSE), and `-h` (alone) prints a help message. Mark5A must *not* be running. *This program erases all data on SS disks.* SSErase is useful to test the write-ability of disks and to prepare disks to be recorded. If a disk pack is write protected, then SSErase asks permission to remove this protection as is necessary to erase.

If *c* is 1 (TRUE), then SSErase also goes through the long conditioning process on whatever disks it can access, up to 16 at a time in both banks. If *m* is set to 0, then debug prints about once a minute during conditioning to show what's happening.

Conditioning disks is recommended before recording especially if they are to be recorded at or near their maximum data rate. Conditioning amounts to a write-read-write cycle through the whole set of disks, but SSErase does this in two rather than three passes. Printed with debuggery and counting down twice is the number of bytes per bus. With an 8-disk pack, for example, this count starts at a quarter of the total capacity, which is twice the capacity of a single disk.

Conditioning one 120-Gbyte disk takes about 101 minutes; two 120-Gbyte disks (as two masters), 103 minutes; four 120-Gbyte disks (as four masters), 111 minutes; eight 120-Gbyte disks (an eight pack), 157 minutes; and sixteen 120-Gbyte disks (two eight packs), 278 minutes. 200-Gbyte disks take about $^{5}/_{3}^{rds}$ as long—no surprise. Conditioning one 200-Gbyte disk takes about 160 minutes; eight 200-Gbyte disks, 286 minutes; and sixteen 200-Gbyte disks, 465 minutes. These times are approximate and for the ideal case, but, if any of the disks has a problem, then times can be much longer. After conditioning two disk packs at a time (i.e., more than eight disks), or in any case where you change the disks into a different configuration, then you should also SSErase (without -c 1) each disk pack separately before recording. This takes only a few extra seconds.

`SSReset`

This program performs an XLRCardReset(), which often helps extricate the system from a no-fair state. Use `SSReset` *before starting* `Mark5A`.

`DirList [ -m m ] [ -f filename ] [ -h ]`

where *m* (msglev) is the debug message level, range -1 to 3, default 1, *filename* sets the name of the Mark-5 directory file, default /var/dir/Mark5A, and -h (alone) prints a help message. DirList reads the Mark-5 directory and lists the contents including the starting and ending byte numbers of all completed scans. DirList can run simultaneously with Mark5A, in which case the listing will be up to date except for any scan being recorded at the time. If Mark5A is dismounted (i.e., no disk pack active), then the listing will be appropriate for the SS disk(s) that were in place before the dismount. Or if Mark5A is not running, then the listing will be appropriate for whatever was happening last time Mark5A was running. This is equivalent to saying that DirList does not read SS disks, instead it reads whatever Mark5A has written into /var/dir/Mark5A or *filename*.

Following are notes about two stand-alone programs intended to exchange data with Mark-5 machines and primarily intended to run on computers other than Mark-5 machines. The source code contains hints for compiling on Linux and HP-UX.

`Net2file [ filename ]`

accepts a connection from a Mark-5 machine and writes the received data to *filename* or to save.data if *filename* is blank. This file will be created if necessary or appended. Start Net2file *first*, then command disc2net or in2net in the Mark-5 machine. Monitor progress with

`ls -l filename`

but this will lag the actual progress because of buffering. Net2file will end when the Mark-5 machine disconnects or with <Ctrl>C, after which *filename* will be ready to use.

`File2net machine [ filename [ startbyte [ endbyte ] ] ]`

sends a file or part of a file to a Mark-5 machine. Command net2disc or net2out in the Mark-5 machine *first*, then start File2net. *Filename* defaults to save.data, *startbyte* defaults to 0, and *endbyte* defaults to the end of the file. File2net ends when the prescribed transfer is done.

`XSVF [ -m m ] [ -f filename ] [ -h ]`

(formerly tstXSVF) where *m* (msglev) is the debug message level, range -1 to 3, default 1, *filename* sets the name of the Xilinx xsvf file to be read, default test.xsvf, and -h (alone) prints a help message. This program transfers the prescribed xsvf file to the PROMs on the Mark-5B I/O board to program the Xilinx programmable gate array (PGA). There will be four choices of xsvf file: DIM or DOM and each with or without phase cal. XSVF takes about six minutes to finish normally. These files and this program are still under construction.

Revised: 2005 August 19, JAB