Making Your Very Own Sprite Kernel

Last Revised: 2/28/92

1. Background Information

1.1. Kernel Sources

The kernel sources are maintained by a home-grown source control system called **scvs**. The scvs cheat sheet (available in `/sprite/admin/howto/scvs`) has lots of information on how to use scvs. The kernel sources are divided up into 32 different *modules*. A list of these modules can be found in `/sprite/src/kernel/Modules`. Read-only copies of the kernel modules can be found in subdirectories of `/sprite/src/kernel`.

1.2. Installed Object Files

The read-only copies of the kernel sources are periodically compiled, and the resulting object files are stored in `/sprite/src/kernel/`*machine*`.md`. These object files, can be linked together to build a kernel.

1.3. Installed Kernels

The "official" Sprite kernels are built in `/sprite/src/kernel/sprite`. The Makefile in that directory links together all of the installed object files to produce an official kernel.

2. Making Your Own Kernel

2.1. Your Personal Build Directory

Sprite kernel developers have their own personal build directories in `/sprite/src/kernel`. The name of the directory is usually the same as their login name. To make your own kernel build directory go to `/sprite/src/kernel` and type **update newuser** *login*, where *login* is your login name. (By the way, the **update** program is really useful so you might want to check it out if you don't already know about it.) Once the update completes go to your build directory and follow the instructions in the README file.

2.2. Modifying a Module

The **scvs** cheat sheet outlines how to make your own copy of a kernel modules and make modifications. To summarize, go to the `src` subdirectory of your kernel build directory and type **scvs co** *module*. Once you have your copy you need to create a Makefile for it. Go to the copy and type **mkmf**. This will create a Makefile for you. If you add new files, or remove existing ones, then you will need to rerun **mkmf**. Tag files are created by typing **pmake tags** once you have a Makefile.

2.3. Building a Kernel

Kernels are built by going to your kernel build directory and typing **pmake** *machine*, where *machine* is the type of kernel that you want to build, e.g. ds5000. If you do not modify the object file list in your Makefile your kernel will be built out of the installed object files. If you want to use your own copy of a module then you need to modify the Makefile (it has comments).

There are few miscellaneous object files that are necessary to build a kernel. The first is version.o, which contains the kernel version string. The second is mainHook.o. This contains random variable definitions and strange routines. I personally avoid modifying it. There are some variables that it may be useful to modify (like one that produces verbose information during the boot). See the mainHook.c files for details.

3. Booting Your Kernel

Your kernel will have the same name as your login. To boot it, replace the "sprite" or "new" in the normal boot command for the machine with your kernel name. For example, if your login name was **foo** and you wanted to boot a ds5000 with your kernel you would type **boot 2/mop/fooR.**

4. Debugging Your Kernel

You need to machines to debug a kernel; one runs the debugger and the other runs your kernel. To debug a machine it must be running the portion of the kernel that handles packets from the debugger running on the other machine (we refer to a machine in this state as "in the debugger"). There are a number of ways of getting the machine into the debugger. One is to have a bug in the kernel. Another is to type **l1-d** at the console, and a third is to type **kmsg -d** *hostname* from another machine. Once the machine is in the debugger you need to determine which kernel it is running. Either type **tail** `/hosts/`*hostname*`/boottimes`*, or type* **kmsg -v** *hostname*. Go to your kernel build directory. Do **strings** *kernel* **| grep VERSION** on your kernel and make sure it corresponds with the kernel that the machine was running. If it does, then type **Kgdb** *kernel*. Once you get a debugger prompt type **attach** *hostname*. It should print a message about dumping out the syslog, followed by a stack backtrace.