CPU "owns" the controller. The backup path is not used until the primary path fails or the system operator manually switches to it (a so-called *primary switch*).

Disks are a particularly sensitive issue because many components could fail. It could be a disk itself, the physical connection (cable) to the disk, the disk controller, the I/O bus, or the CPU to which it is connected. As a result, in addition to the dual-ported controllers, each disk is physically duplicated—at least in theory—and it is also dual-ported and connected to two different controllers, both connected to the same two CPUs The restriction remains that only one CPU can access each controller at any one time, but it is possible for one of the CPUs to own one of the controllers and the other CPU to own the other controller. This is also desirable from a performance point of view.

Figure 8-1 shows a typical configuration: as the gray highlighted paths indicate, the I/O process for the system disk `$SYSTEM` accesses it via CPU 0 and the first disk controller, while the I/O process for another disk `$DATA`, connected to the same two controllers, accesses the disk via CPU 1 and the second disk controller. CPU 0 "owns" the first controller, while CPU 1 "owns" the second controller. If CPU 0 were to fail, the backup I/O process for `$SYSTEM` in CPU 1 would take over and take ownership of the controller, and then continue processing. If the second disk controller were to fail, the I/O process for `$DATA` would not be able to use the first disk controller, since it is owned by CPU 0, so the I/O process would first do a primary switch, after which the primary I/O process would be running in CPU 0. It would then access `$DATA` by the same path as `$SYSTEM`.

That's the theory, anyway. In practice, disks and drives are expensive, and many people run at least some of their disks in degraded mode, without duplicating the drive hardware. This works as well as you would expect, but of course there is no longer any further fault tolerance: effectively, one of the disks has already failed.

Process Structure

Guardian is a microkernel system: apart from the low-level interrupt handlers (a single procedure, `IOINTERRUPT`) and some very low-level code, all the system services are performed by system processes which run in system code and data space.

The more important processes are:

- The *system monitor*, PID 0 in each CPU, which is responsible for starting and stopping other processes and for miscellaneous tasks such as returning status information, generating hardware error messages, and maintaining the time of day.
- The *memory manager*, PID 1 in each CPU, which is responsible for I/O for the virtual memory system.
- The I/O processes, which are responsible for controlling I/O devices. All access to I/O devices from anywhere in the system goes via its dedicated I/O process. The I/O controllers