PHILIPS

PHILIPS INTERACTIVE MEDIA OF AMERICA

Technical Note #72

Passing Open Paths Between Processes

Charles Golvin

December 2, 1991

CD-I applications may employ more than one process. It may be required that these processes share a path to an open device. This note describes the technique for passing open paths between processes, and for determining the type of CD-I device associated with a path number.

Passing Open Paths Between Processes

In any CD-I application, part of the initialization process is to open paths to the various devices that will be used. These include, but are not limited to, the video device, audio device, and the disc. In a multi-process application, the processes that are started by the initial process generally require access to one or more of these devices. In this case, it is desirable that the parent process pass the open paths to the child processes. This saves time and, in the case of disc files, memory.

The preferred method for starting a new process is to use the OS-9 system call **os9exec()**, which then uses one of the following calls to create the new process:

- **os9fork()**
- **os9forkc()**
- **chain()**
- **chainc()**

The difference between "forking" and "chaining" depends on whether the parent process remains active or disappears. The difference between **os9fork()** and **os9forkc()** and between **chain()** and **chainc()** consists of an extra parameter on the latter call that specifies how many paths are to be passed to the created process.

If the parent process uses **chainc()** or **os9forkc()** and specifies a number of paths to be inherited by the child, what paths are passed? All the paths whose path number is less than the number specified. That is, if **N** paths are specified, paths numbered **0** through **N-1** are passed.

It is the responsibility of the parent process to pass the correct paths to its child process. By the same token, it is the responsibility of the child process to ascertain which path belongs to which device. Fortunately for child processes everywhere, the *CD-I Full Functional Specification* specifies how to determine to which type of device a given path number pertains.

The **I$GetStat** call with function code **SS_Opt** (or its C language equivalent **_gs_opt()**) returns the options fields of the path descriptor belonging to a process's path number. The options fields for CDFM devices are given in chapter VII, pp. 226-229 of the specification; for UCM devices consult chapter VII, pp. 242-245.

```
#define CDFM_CLASS              5
#define UCM_CLASS               6
#define DEVICE_CLASS_OFFSET     0
#define PD_CDFC                 1
#define CD_DEVICE               0
#define AUDIO_DEVICE            1
#define PD_FC                   28
#define KEYBOARD                1
#define POINTER                 2
#define VIDEO                   4
#define SYSERR                  (-1)
#define BUFFER_SIZE             256

  char abuf[ BUFFER_SIZE ];
  int pthidx = 0;

  while ( _gs_opt( pthidx, abuf ) != SYSERR )
  {
    switch( abuf[ DEVICE_CLASS_OFFSET ] )
    {
      case CDFM_CLASS.                  /* This is a path to a CDFM device */

        switch ( abuf[ PD_CDFC ] )
        {
          case CD_DEVICE.               /*A path to the CD device */
            break;
          case AUDIO_DEVICE.            /*A path to the AUDIO device */
            break;
          default:                      /*An unknown CDFM path type */
            break;
        }
        break;

      case UCM_CLASS                    /* A path to a UCM device */

        if ( abuf[ PD_FC ] & VIDEO )    /* A path to the video device */
          .
        else if ( abuf[ PD_FC ] & POINTER )   /* A path to the pointer device */
          .
        else if ( abuf[ PD_FC ] & KEYBOARD )  /* A path to the keyboard device */
          .
        else                            /* An unknown UCM path type */
          .
        break;

      default:                          /* This is some other type of path, perhaps SCF */


        break;
    }
  }
```