Using the Process Manager Information for scheduling all processes

The **Process Manager** assigns a process serial number to each open application (or desk accessory, if it is not opened in the context of an application). The process serial number is unique to each process on the local machine and is valid for a single boot of the machine. You can use the process serial number to specify a particular process for most of the **Process Manager** routines. You can use the process serial number returned from the **GetCurrentProcess** , **GetNextProcess** , **GetFrontProcess** , and **LaunchApplication** functions in other **Process Manager** routines.

Opening or Printing Files Based on Finder Information

When a user opens or prints a file from the Finder, the Finder uses the **Process Manager** to launch the application that created the file. The Finder sets up the information your application can use to determine which files to open or print. The Finder information includes a list of files to open or print.

In version 7.0, applications that support high-level events (that is, that have the isHighLevelEventAware flag set in the 'SIZE' resource) receive the Finder information through Apple events. The **Apple Event Manager** describes how your application processes Apple events to open or print files.

Applications that do not support high-level events can use the **CountAppFiles** , **GetAppFiles** , and **ClrAppFiles** routines or the **GetAppParms** routine to get the Finder information. See the section called, **Segment Loader** for information on these routines.

Getting Information About Other Processes

You can use the **GetNextProcess** , **GetFrontProcess** , or **GetCurrentProcess** functions to get the process serial number of a process. The **GetCurrentProcess** function returns the process serial number of the process that is currently executing. The current process is the process whose A5 world is currently valid; this process can be in the background or foreground. The **GetFrontProcess** function returns the process serial number of the foreground process. For example, if your process is running in the background, you can use **GetFrontProcess** to determine which process is in the foreground.

The **Process Manager** maintains a list of all open processes. You can specify the process serial number of a process currently in the list and use **GetNextProcess** to get the process serial number of the next process in the list. The interpretation of the value of a process serial number and the order of the list of processes is internal to the **Process Manager**.

When specifying a particular process, use only a process serial number returned by a high-level event, **Process Manager** routine, or constants defined by the **Process Manager**. You can use these constants to specify special processes.

```
kNoProcess      //process does not exist
kSystemProcess  //process belongs to OS
kCurrentProcess //the current process
```

In all **Process Manager** routines, the constant kNoProcess refers to a

process that does not exist, the constant kSystemProcess refers to a process belonging to the Operating System, and the constant kCurrentProcess refers to the current process.

To begin enumerating a list of processes, use the **GetNextProcess** function and specify the constant kNoProcess as the parameter to return the process serial number of the first process in the list. You can use the returned process serial number to get the process serial number of the next process in the list. The **GetNextProcess** function returns the constant kNoProcess and the result code procNotFound at the end of the list.

You can also use a process serial number to specify a target application when your application sends a high-level event. See the **Event Manager** for information on how to use a process serial number when your application sends a high-level event.

You can use the **GetProcessInformation** function to obtain information about any process, including your own. For example, for a specified process, you can find the application's name as it appears in the Application menu, the type and signature of the application, the number of bytes in the application partition, the number of free bytes in the application heap, the application that launched the application, and other information.

The **GetProcessInformation** function returns information about the requested process in a process information record. The process information record is defined by the **ProcessInfoRec** data type.

```
// Searching for a specific process

// Assuming inclusion of <MacHeaders>

#include <Processes.h>

Boolean FindAProcess (OSType signature, ProcessSerialNumber
                                        *process,ProcessInfoRec *infoRec,
                                        FSSpecPtr aFSSpecPtr);

Boolean FindAProcess (OSType signature, ProcessSerialNumber *process,
                                        ProcessInfoRec *infoRec, FSSpecPtr
                                        aFSSpecPtr)
{
    process->highLongOfPSN = 0;
    process->lowLongOfPSN = kNoProcess;  // start from the beginning

    infoRec->processInfoLength = sizeof(ProcessInfoRec);
    infoRec->processName = (StringPtr) NewPtr(32);
    infoRec->processAppSpec = aFSSpecPtr;

    while (!GetNextProcess(process)) {

        if ( GetProcessInformation(process, infoRec) ) {
            if ( (infoRec->processType == (long) 'APPL') &&
                 (infoRec->processSignature == signature) )
                return TRUE;                    // found the process
```

```
        }
    } // while

    return FALSE;
}
```

The code example shows how you can use the **GetNextProcess** function with the **GetProcessInformation** function to search the process list for a specific process.

The code searches the process list for the application with the specified signature. For example, you might want to find a specific process in order to send a high-level event to it.