SysEnvirons Get ROM version, machine type, system version, etc.

#include <OSUtils.h> **Operating System Utilities**

```
OSErr     SysEnvirons(verReqNo, theEnv);
short     verReqNo;     desired version of SysEnvRec (1 is safe)
SysEnvRec *theSysEnv;   receives all kinds of info re current machine
          returns       Error Code (0=no error)
```

SysEnvirons is no longer a recommended way of obtaining information about the environment in which your code will run. **Gestalt** is now available as glue in development environments so that there is no need to call **Environs** or **SysEnvirons**, regardless of what machine or operating system you are running on.

SysEnvirons fills a structure with information about the computer and system on which the caller is executing.

verReqNo identifies which version of the SysEnvRec you wish to obtain. Use the constant curSysEnvVers (currently version 2) to obtain the basic 18-byte version described in the SysEnvRec topic.

theSysEnv is the address of an 18-byte SysEnvRec structure. Upon return, it contains all the system environment information. See SysEnvRec for details of the return values.

Returns: an OSErr; an integer Error Code. It will be one of:

noErr	(0)	No error
envNotPresent	(-5500)	_SysEnvirons trap not present
envBadVers	(-5501)	negative number used in verReqNo
envVersTooBig	(-5502)	verReqNo version not handled by this system

---

Notes: **SysEnvirons** obtains the ROM version number, such as the CPU type, whether or not color Quickdraw is present, keyboard type, and so forth.

Macintosh Technote #129 warns that values may be added to the acceptable set of values for any field of the SysEnvRec without warning. Therefore, if you are calling **SysEnvirons** you should be prepared to handle unexpected values.

The constant curSysEnvVers contains the current version of **SysEnvirons**. By using this constant rather than a hard coded version number you can reduce the changes that may need to be made to your source as **SysEnvirons** evolves. However, you should be prepared to handle unexpected values and not make assumptions about functionality based on current expectations. It is much safer to test for specific functionality with **Gestalt** than rely on the future contents of **SysEnvirons**.

In future systems, the SysEnvRec structure may get longer, so the following mechanism can be used to get the most up-to-date version of this record:

- Set *verReqNo* to a value greater than curSysEnvVers (the SysEnvRec version number you suspect might not be available).
- Set *theSysEnv* to point to the larger structure (i.e., some future

structure longer than 18 bytes).

- Call **SysEnvirons**. Upon return, if version number *verReqNo* is not available, the return OSErr will be an Error Code of envVersTooBig, the first field of the structure (SysEnvRec.environsVersion) will be less than the value you requested (indicating you received less information than you wanted), and the structure at *theSysEnv* will contain only the information from the older version of the structure.

If you call **SysEnvirons** on a Mac that does not support it, the glue routine provided by your compiler will fill-in what ever information it can, and return an Error Code of envNotPresent.

Inside Macintosh (V, page 5) states that "all of the Toolbox Managers must be initialized before calling **SysEnvirons**." Technical Note #129 points out that this isn't correct: startup documents (for example) can call **SysEnvirons** before Toolbox initialization, though the SysEnvRec.atDrvrVersNum will return 0 if the AppleTalk drivers are not initialized. The system version, machine type, processor type, and other key data are returned normally.

Example

```
#include <OSUtils.h>
#include <StdIO.h>

short           errCode;
SysEnvRec       ser;

errCode = SysEnvirons( curSysEnvVers, &ser );
if ( errCode < 0 ){
        printf("System older than 4.1\n");
        printf( "The machine has Color Quickdraw%s\n", ser.hasColorQD ? "." :
        "... NOT!");
}
```