Technical Note TN2006

MP-Safe Routines

CONTENTS

This technote lists all routines callable from MP tasks under Multiprocessing Services 2.0 and higher. This list of routines is larger than you might expect, and is growing steadily. You may find some pleasant surprises in the latest list.

This Note is directed at all developers with a compute-bound or I/O-bound application. It is now possible to do this type of work from MP tasks, with potential performance and power-saving benefits with both Mac OS 9 and Mac OS X.

Updated: [Dec 21 2000]

Introduction

With the introduction of Multiprocessing Services 2.0 (Mac OS 8.6) and renewed availability of multiprocessor hardware, many developers are reconsidering whether to adopt the preemptive and multiprocessor-aware threading provided by Multiprocessing Services. Historically, the biggest obstacle to the adoption of Multiprocessing Services has been the limited set of system routines that are callable from MP tasks. However, this set is steadily increasing and is now sufficient for many real-world applications. This technote provides a list of MP-safe system routines, and when they became MP-safe.

> **IMPORTANT:**
> The MP tasks provided by Multiprocessing Services 2.0 are not just for use on MP systems. They provide a general purpose preemptive threading mechanism on all computers, even those with a single CPU. If your application needs preemptive threads, you should consider using MP tasks. Specifically, if your code is either compute-bound or I/O-bound, it's likely that adopting MP tasks will yield some combination of performance benefits, power savings, and maintainability improvements.

> **WARNING:**
> This technote discusses Multiprocessing Services 2.0 (Mac OS 8.6) and higher exclusively. It does not cover any issues related to Multiprocessing Services 1.x.

Most of the APIs listed in this technote are part of Carbon. Accessing an API through CarbonLib on traditional Mac OS does not affect its MP-safeness: if the routine is MP-safe for an InterfaceLib-based application, it will also be MP-safe for a CarbonLib-based one. All of the MP-safe routines listed here are MP-safe on all versions of Mac OS X.

This technote concentrates on routines that are callable from preemptive tasks on both Mac OS 9 and Mac OS X. Many Mac OS X routines that are not part of Carbon, for example, the BSD file and network APIs, are MP-safe but are not described here.

The list of MP-safe routines described in this technote supplants the list given in the Preemptive Task-Safe Mac OS System Software Functions section of Adding Multitasking Capability to Applications Using Multiprocessing Services.

Back to top

MP-Safe Routines by Manager

MP-Safe Routines by Manager

This section lists all MP-safe routines, grouped by the manager that implements the routine. In cases where the description refers to header files, it refers to Universal Interfaces 3.3.2. This list is accurate as of Mac OS 9.0.4.

Debugging

With MacsBug (version 6.6.3 or greater) installed, it is safe to call `Debugger`, `DebugStr`, and `debugstr` from MP tasks.

Multiprocessing Services

All Multiprocessing Services routines are callable from MP tasks. Specifically, all the non-deprecated routine declared in "Multiprocessing.h" are callable from MP tasks.

See Adding Multitasking Capability to Applications Using Multiprocessing Services for more details on these routines.

Memory Manager

The following Memory Manager routines are MP-safe: `BlockMove`, `BlockMoveData`, `BlockMoveUncached`, `BlockMoveDataUncached`, `BlockZero`, and `BlockZeroUncached`.

See Inside Macintosh: Memory and Designing PCI Cards and Drivers for Power Macintosh Computers for more details on these routines.

Driver Services

All of the atomic operations originally introduced in DriverServicesLib (and exported by InterfaceLib since Mac OS 8.5) are MP-safe. Specifically, all of the routines declared in "DriverSynchronization.h" are MP-safe.

> **WARNING:**
> Do not use the PowerPC instructions Load Reserved (`lwarx`) and Store Conditional (`stwcx`) to implement atomicity in your preemptively threaded application. As described in DTS Technote 1137 Disabling Interrupts on the Traditional Mac OS, these instructions are non-portable and are tricky to use correctly across the full spectrum of PowerPC implementations.

> **WARNING:**
> The DriverServicesLib queue manipulation routines (for example, `PBEnqueue`) are not MP-safe.

Most of the DriverServicesLib time measurement routines are MP-safe. This includes the routines `UpTime`, `AbsoluteToNanoseconds`, `AbsoluteToDuration`, `NanosecondsToAbsolute`, `DurationToAbsolute`, `AddAbsoluteToAbsolute`, `SubAbsoluteFromAbsolute`, `AddNanosecondsToAbsolute`, `AddDurationToAbsolute`, `SubNanosecondsFromAbsolute`, `SubDurationFromAbsolute`, `AbsoluteDeltaToNanoseconds`, `AbsoluteDeltaToDuration`, `DurationToNanoseconds`, and `NanosecondsToDuration`. The routine `DelayForHardware` is MP-safe but you should use the routine `MPDelayUntil` instead, because it yields better CPU utilitization and power savings.

All of the DriverServicesLib string manipulation routines are MP-safe. This includes the routines `CStrCopy`, `PStrCopy`, `CStrNCopy`, `PStrNCopy`, `CStrCat`, `PStrCat`, `CStrNCat`, `PStrNCat`, `PStrToCStr`, `CStrToPStr`, `CStrCmp`, `PStrCmp`, `CStrNCmp`, `PStrNCmp`, `CStrLen`, and `PStrLen`.

The `DriverServicesLib` routine `BlockCopy` is MP-safe.

See Designing PCI Cards and Drivers for Power Macintosh Computers for more details on these routines.

File Manager

In Mac OS 9.0 and above, it is safe to call the bulk of the File Manager API synchronously from an MP task. See the Preemptive Task-Safe Mac OS System Software Functions section of Adding Multitasking Capability to Applications Using Multiprocessing Services.

> **IMPORTANT:**
> See the section CarbonLib and MP for an important caveat.

> **WARNING:**
> Rather than test for a specific system software version, your application should test for this functionality by

calling `Gestalt` with the `gestaltMPCallableAPIsAttr` selector and checking the `gestaltMPFileManager` bit in the response.

The File Manager is described in detail by a number of documents on the developer web site.

Device Manager

In Mac OS 9.0 and above it is safe to call core Device Manager routines synchronously from an MP task. See the Preemptive Task-Safe Mac OS System Software Functions section of Adding Multitasking Capability to Applications Using Multiprocessing Services.

IMPORTANT:
See the section CarbonLib and MP for an important caveat.

WARNING:
Rather than test for a specific system software version, your application should test for this functionality by calling `Gestalt` with the `gestaltMPCallableAPIsAttr` selector and checking the `gestaltMPDeviceManager` bit in the response.

See Inside Macintosh: Devices for more details on these routines.

Process Manager

In Mac OS 9.0 and above it is safe to call the Process Manager routine `WakeUpProcess` from an MP task.

IMPORTANT:
See the section CarbonLib and MP for an important caveat.

IMPORTANT:
This feature is very important if you need to communicate between MP tasks and your main thread. Imagine you have a network server that does all of its core work in MP tasks. The main thread's only job is to start the MP tasks and then block waiting for user interface events. It waits for events by calling `WaitNextEvent` with a very large sleep time. This makes it friendly to other users of the CPU and maximizes power savings when the server is not busy.

Now let's imagine that the server supports remote administration and some remote administration requests must be performed by the main thread. A good example of this is a request to quit the server. The MP task handling that request must wake up the main thread, even though the main thread is blocked inside a call to `WaitNextEvent`. It can do this by calling `WakeUpProcess`.

`WakeUpProcess` is documented in Inside Macintosh: Processes.

Deferred Task Manager

In Mac OS 9.0 and above it is safe to call the Deferred Task Manager routine `DTInstall` from an MP task.

IMPORTANT:
See the section CarbonLib and MP for an important caveat.

IMPORTANT:
Do not underestimate the utility of this facility. It provides a general purpose low-latency communications mechanism between MP tasks and any "blue" code that can be called at interrupt time. If you need to call an interrupt-safe (but not MP-safe) routine from an MP task, you can use `DTInstall` to schedule a deferred task to do the work. The OTMP library (discussed in the next section) uses this technique extensively.

`DTInstall` is documented in Inside Macintosh: Processes.

Open Transport

Many Open Transport utility routines are MP-safe. These routines are summarized in Tables 1 through 11.

Table 1. MP-safe OT debugging utilities

Routine	Comment

OTDebugStr	
OTDebugBreak	macro that calls OTDebugStr
OTDebugTest	macro that calls OTDebugStr
OTAssert	macro that calls OTDebugStr
OTDebugBreak2	macro that calls OTDebugStr
OTDebugTest2	macro that calls OTDebugStr

Table 2. OT port reference manipulators

Routine	Comment
OTCreatePortRef	
OTGetDeviceTypeFromPortRef	
OTGetBusTypeFromPortRef	
OTGetSlotFromPortRef	
OTSetDeviceTypeInPortRef	
OTSetBusTypeInPortRef	
OTCreateNuBusPortRef	macro that calls OTCreatePortRef
OTCreatePCIPortRef	macro that calls OTCreatePortRef
OTCreatePCCardPortRef	macro that calls OTCreatePortRef

Table 3. OT buffer manipulation

Routine	Comment
OTInitBufferInfo	macro
OTNextLookupBuffer	macro
OTNextOption	
OTFindOption	
OPT_NEXTHDR	macro
datamsg	macro
OTBufferDataSize	
OTReadBuffer	
OTReleaseBuffer	
StoreIntoNetbuf	
StoreMsgIntoNetbuf	

Table 4. OT memory and string utilities

Routine	Comment
OTMemcpy	
OTMemcmp	
OTMemmove	
OTMemzero	
OTMemset	
OTStrLength	
OTStrCopy	
OTStrCat	
OTStrEqual	

Table 5. OT list utilities

Routine	Comment
OTGetLinkObject	macro
OTLIFOEnqueue	
OTLIFODequeue	
OTLIFOStealList	
OTReverseList	still need mutual exclusion, see below
OTAddFirst	still need mutual exclusion, see below
OTAddLast	still need mutual exclusion, see below
OTRemoveFirst	still need mutual exclusion, see below
OTRemoveLast	still need mutual exclusion, see below
OTGetFirst	still need mutual exclusion, see below
OTGetLast	still need mutual exclusion, see below
OTIsInList	still need mutual exclusion, see below
OTFindLink	still need mutual exclusion, see below
OTRemoveLink	still need mutual exclusion, see below
OTFindAndRemoveLink	still need mutual exclusion, see below
OTGetIndexedLink	still need mutual exclusion, see below
OTEnqueue	
OTDequeue	

Table 6. OT atomic utilities

Routine	Comment
OTAtomicSetBit	
OTAtomicClearBit	
OTAtomicTestBit	
OTCompareAndSwapPtr	
OTCompareAndSwap32	
OTCompareAndSwap16	
OTCompareAndSwap8	
OTAtomicAdd32	
OTAtomicAdd16	
OTAtomicAdd8	
OTClearLock	macro
OTAcquireLock	macro
OTSetFirstClearBit	
OTClearBit	
OTSetBit	
OTTestBit	

Table 7. TCP/IP utilities

Routine	Comment
SET_TOS	macro
OTInitInetAddress	
OTInitDNSAddress	
OTInetStringToHost	
OTInetHostToString	

Table 8. AppleTalk utilities

Routine	Comment
IsAppleTalkEvent	macro
OTCopyDDPAddress	
OTInitDDPAddress	
OTInitNBPAddress	
OTInitDDPNBPAddress	
OTCompareDDPAddresses	
OTInitNBPEntity	
OTGetNBPEntityLengthAsAddress	
OTSetAddressFromNBPEntity	
OTSetAddressFromNBPString	
OTSetNBPEntityFromAddress	
OTSetNBPName	
OTSetNBPType	
OTSetNBPZone	
OTExtractNBPName	
OTExtractNBPType	
OTExtractNBPZone	

Table 9. Ethernet utilities

Routine	Comment
OTCompare48BitAddresses	macro
OTCopy48BitAddress	macro
OTClear48BitAddress	macro
OTCompare8022SNAP	macro
OTCopy8022SNAP	macro
OTIs48BitBroadcastAddress	macro
OTSet48BitBroadcastAddress	macro
OTIs48BitZeroAddress	macro

Table 10. Serial utilities

Routine	Comment
OTSerialHandshakeData	macro/inline
OTSerialSetErrorCharacter	macro/inline
OTSerialSetErrorCharacterWithAlternate	macro/inline

Table 11. OT advanced utilities

Routine	Comment
OTSetFirstClearBit	
OTClearBit	
OTSetBit	
OTTestBit	
OTCalculateHashListMemoryNeeds	
OTInitHashList	
OTAddToHashList	still need mutual exclusion, see below
OTRemoveLinkFromHashList	still need mutual exclusion, see below
OTIsInHashList	still need mutual exclusion, see below
OTFindInHashList	still need mutual exclusion, see below
OTRemoveFromHashList	still need mutual exclusion, see below
OTGetRandomSeed	
OTGetRandomNumber	
OTInitGate	
OTEnterGate	
OTLeaveGate	

Some of the Open Transport utility routines (those that manipulate OT lists and hash tables) are safe to call from an MP task, but the underlying data structure is not re-entrant. These routines are tagged with the text "still need mutual exclusion" in the tables above. If you call these routines from preemptive code, you must be sure to synchronize your operations on any given object such that the object is not corrupted by re-entrant calls. One way to do this is to have each MP task that operates on an object enter a critical section before calling OT and leave the critical section once OT returns.

The Open Transport routines listed above are inherently MP-safe, but do not let you send or received network data. They are useful utilities only. If you wish to send or receive network data from an MP task, you must use the OTMP library. Table 12 shows the appropriate OTMP routine to call for any given OT routine.

The OTMP library is available as DTS sample code. It requires Mac OS 9.0 or later. See the documentation that comes with the library for more details on how to use it.

Table 12. Routines supported by the OTMP library

Routine Needed	OTMP Routine to Call
InitOpenTransportInContext	use InitOpenTransportMPXInContext
InitOpenTransport	use InitOpenTransportMPXInContext
CloseOpenTransportInContext	use CloseOpenTransportMPXInContext
CloseOpenTransport	use CloseOpenTransportMPXInContext
OTOpenEndpointInContext	use OTMPXOpenEndpointQInContext
OTOpenEndpoint	use OTMPXOpenEndpointQInContext
OTCloseProvider	use OTMPXCloseProvider
OTIoctl	use OTMPXIoctl
OTCancelSynchronousCalls	use OTMPXCancelSynchronousCalls
OTGetEndpointInfo	use OTMPXGetEndpointInfo
OTGetEndpointState	use OTMPXGetEndpointState
OTLook	use OTMPXLook
OTCountDataBytes	use OTMPXCountDataBytes
OTGetProtAddress	use OTMPXGetProtAddress
OTResolveAddress	use OTMPXResolveAddress

OTBind	use OTMPXBind
OTUnbind	use OTMPXUnbind
OTConnect	use OTMPXConnect
OTListen	use OTMPXListen
OTAccept	use OTMPXAccept
OTSndDisconnect	use OTMPXSndDisconnect
OTRcvDisconnect	use OTMPXRcvDisconnect
OTSndOrderlyDisconnect	use OTMPXSndOrderlyDisconnect
OTRcvOrderlyDisconnect	use OTMPXRcvOrderlyDisconnect
OTOptionManagement	use OTMPXOptionManagement
OTRcv	use OTMPXRcv
OTSnd	use OTMPXSnd
OTSndUData	use OTMPXSndUData
OTRcvUData	use OTMPXRcvUData
OTRcvUDErr	use OTMPXRcvUDErr

IMPORTANT:
See the section CarbonLib and MP for an important caveat.

Note:
The OTMP library does not provide an equivalent for the OTInetStringToAddress routine. This is a deliberate omission. Most users of that routine would be better served by using AF_DNS addresses. See the discussion of addressing in Inside Macintosh: Networking with Open Transport and the DTS sample code OTSimpleDownloadHTTP for details.

Note:
The techniques used by the OTMP library to layer a synchronous MP-safe API on top of an asynchronous interrupt-safe API are applicable to other problem domains, such as most traditional Mac OS I/O services.

Inside Macintosh: Networking with Open Transport is the core API reference for Open Transport.

Back to top

CarbonLib and MP

Due to a problem in current versions of the CarbonLib extension [2563553], many of the routines that were are documented as MP-safe in Mac OS 9.0 and above are not MP-safe if any version of CarbonLib prior to version 1.2.5 is installed. This problem affects both InterfaceLib and CarbonLib applications. If the user has run a CarbonLib application since the system was booted, you will find that calling these routines from an MP task will crash the system.

IMPORTANT:
If your application uses the Multiprocessing Library routines, then you should ensure CarbonLib 1.2.5 (or later) is installed.

This problem affects the following system services as they are described in this technote:

- File Manager
- Device Manager
- Process Manager
- Deferred Task Manager
- OTMP

The OTMP sample code library contains an example of how you can detect the presence of a CarbonLib with this problem.

Back to top

Summary

The set of MP-safe system routines has been steadily growing since the release of Multiprocessing Services 2.0. With Mac OS 8.6, only compute-bound applications could use MP tasks in a useful way. With Mac OS 9.0 and OTMP, the set of MP-safe routines is now large enough to support most I/O-bound applications. The time has come to think about using MP tasks in your application, and exploit the benefits available on both Mac OS 9 and X.

Back to top

References

Adding Multitasking Capability to Applications Using Multiprocessing Services

Inside Macintosh: Networking with Open Transport

Inside Macintosh: Processes

Inside Macintosh: Memory

Inside Macintosh: Devices

Designing PCI Cards and Drivers for Power Macintosh Computers

MacsBug releases

DTS Technote 1104 Interrupt-Safe Routines

DTS Technote 1137 Disabling Interrupts on the Traditional Mac OS

DTS sample code OTSimpleDownloadHTTP

Back to top

Downloadables

Acrobat version of this Note (124K). Download

Back to top
