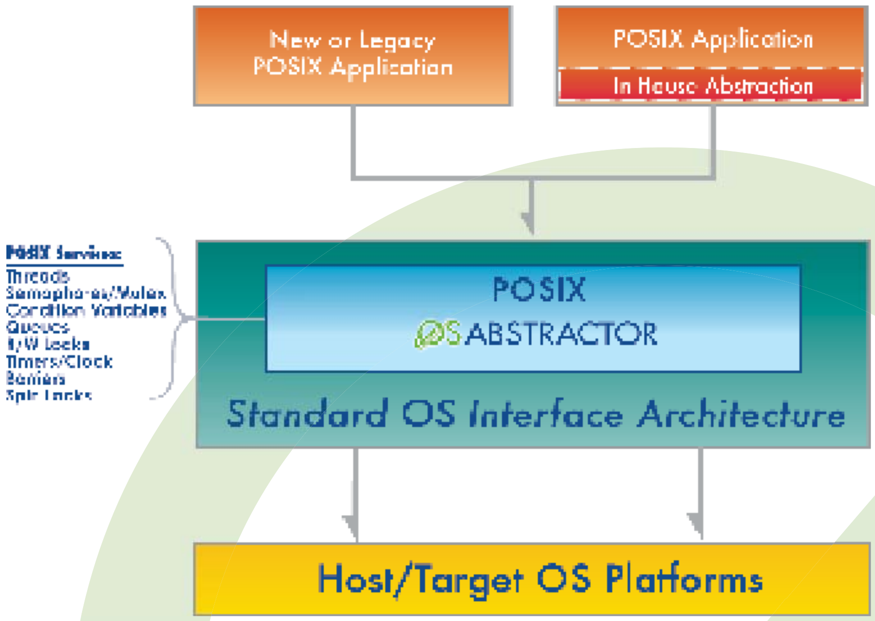
OS ABSTRACTOR®

POSIX OS Abstractor

Write Portable Code - Protect Your Investment

OS Abstractor provides you a robust and standard OS interface architecture for flexible application development while eliminating the risks associated with selecting an OS and dependency on a single vendor. OS Abstractor makes your application adapt to multiple operating system platforms with a standard OS interface, thereby reducing cost associated with code maintenance and learning multiple operating systems. POSIX OS Abstractor enhances the BASE OS Abstractor standard OS interface architecture with the addition of optimized non-proprietary and industry standard POSIX APIs to facilitate using open source POSIX/Linux in your design.



POSIX OS Abstractor Highlights

- Vendor independent and industry standards based solution protects your code investment and knowledge-base
- Leverage re-usable open source POSIX/Linux code to efficiently add feature rich services economically regardless of the underlying OS
- Get to market faster with compelling open-source applications and content in your design
- Tap into the large talent pool of engineers with POSIX/Linux experience
- Offers a high level of code re-usability across many supported POSIX and non-POSIX OS
- Scalability at component & individual feature levels to lower memory foot-print
- Offers POSIX OS functionality to enable complex applications to work on a single-memory address based real-time OS
- Set limit on individual application's heap memory
- Highly optimized for speed and memory footprint for each specific target OS
- Includes BASE OS Abstractor APIs for added flexibility in your development (refer to BASE OS Abstractor Datasheet)
- Easily connects to your in-house abstraction solution
- Easily extended to support your proprietary OS to enable POSIX compliance and/or to re-use open source solutions
- Offered royalty free and with source code

Using POSIX OS Abstractor

POSIX OS Abstractor is designed for use as a fully scalable C library. Services used inside your application software are extracted from the OS Abstractor libraries and are combined with the other application objects to produce the complete image. This image may be downloaded to the target platform or placed in ROM on the target platform. Application developers need to specify the OS for the application and also include the required OS Abstractor libraries while building the application. Application developers can also select the individual OS Abstractor components that are needed and exclude the ones that are not required.

OS ABSTRACTOR®

POSIX OS Abstractor API Support

The table below lists the POSIX API support offered across various OS platforms. MapuSoft's supported target operating systems include: VxWorks® 5x/6x, Windows® CE/Mobile/XP/Vista, Linux®/RT Linux®, LynxOS®/LynxOS-SE®, MQX®, Solaris®, Unix®, Nucleus®, µITRON®, ThreadX®, T-Kernel®, QNX® and eCOS®. Please note that MapuSoft may provide further support to include additional APIs or operating systems not listed. For a current listing visit http://mapusoft.com/products/offerings or email: info@mapusoft.com

POSIX APIs			
POSIX Threads	***Mutex***	***R/W Locks***	***Condition Variables***
pthread_attr_getinheritsched	pthread_mutex_timedlock	pthread_rwlock_destroy	pthread_cond_destroy
pthread_attr_getschedpolicy	pthread_mutexattr_getprotocol	pthread_rwlock_init	pthread_cond_init
pthread_attr_getscope	pthread_mutexattr_setprotocol	pthread_rwlock_rdlock	pthread_cond_wait
pthread_attr_setinheritsched	pthread_mutex_getprioceiling	pthread_rwlock_tryrdlock	pthread_cond_timedwait
pthread_attr_setschedpolicy	pthread_mutex_setprioceiling	pthread_rwlock_timedrdlock	pthread_cond_signal
pthread_attr_setscope	pthread_mutexattr_getprioceiling	pthread_rwlock_wrlock	pthread_cond_broadcast
pthread_attr_getstackaddr	pthread_mutexattr_setprioceiling	pthread_rwlock_trywrlock	pthread_condattr_destroy
pthread_attr_setstackaddr	pthread_mutexattr_getpshared	pthread_rwlock_timedwrlock	pthread_condattr_init
pthread_attr_getstacksize	pthread_mutexattr_setpshared	pthread_rwlock_unlock	pthread_condattr_getclock
pthread_attr_setstacksize	pthread_mutex_destroy	pthread_rwlockattr_destroy	pthread_condattr_setclock
pthread_attr_destroy	pthread_mutex_init	pthread_rwlockattr_init	pthread_condattr_getpshared
pthread_attr_getdetachstate	pthread_mutex_lock	pthread_rwlockattr_getpshared	pthread_condattr_setpshared
pthread_attr_getschedparam	pthread_mutex_trylock	pthread_rwlockattr_setpshared	
pthread_attr_init	pthread_mutex_unlock		***Device I/O - ANSI***
pthread_attr_setdetachstate	pthread_mutexattr_destroy	***Queues***	OS_Creat
pthread_attr_setschedparam	pthread_mutexattr_init	mq_close	OS_unlink
pthread_attr_getstack	pthread_mutexattr_gettype	mq_open	OS_remove
pthread_attr_setstack	pthread_mutexattr_settype	mq_send	OS_close
pthread_create		mq_receive	OS_open
pthread_setcancelstate	***Semaphores***	mq_timedsend	OS_rename
pthread_cancel	sem_close	mq_timedreceive	OS_read
pthread_testcancel	sem_open	mq_notify	OS_write
pthread_detach	sem_destroy	mq_unlink	OS_ioctl
pthread_equal	sem_init	mq_getattr	OS_lseek
pthread_exit	sem_getvalue	mq_setattr	OS_chdir
pthread_getspecific	sem_wait		OS_getcwd
pthread_join	sem_trywait	***Clock***	OS_getwd
pthread_key_create	sem_timedwait	clock	OS_Printf
pthread_key_delete	sem_post	sleep	OS_Sprintf
pthread_once	sem_unlink	unsleep	
pthread_self		nanosleep	***Spin Locks***
pthread_setcanceltype	***Barriers***	clock_getres	pthread_spin_destroy
pthread_setspecific	pthread_barrierattr_destroy	clock_gettime	pthread_spin_init
pthread_sigmask	pthread_barrierattr_init	clock_settime	pthread_spin_lock
pthread_getschedparam	pthread_barrierattr_getpshared	clock_nanosleep	pthread_spin_trylock
pthread_cleanup_pop	pthread_barrierattr_setpshared	sched_yield	pthread_spin_unlock
pthread_cleanup_push			
pthread_kill			

POSIX APIs

Process		***Miscellaneous***	***Signaling***
posix_spawn	execle	clock_getcpuclockid	signal
posix_spawnp	execlp	glob	sigqueue
posix_spawn_file_actions_addclose	execv	globfree	sigpause
posix_spawn_file_actions_addopen	execvp	getenv	sigsuspend
posix_spawn_file_actions_addup2	getpgrp	setenv	sigwait
posix_spawn_file_actions_init	getppid	shm_unlink	sigwaitinfo
posix_spawn_file_actions_destroy	setsid	times	sigtimedwait
posix_spawnattr_init	exit	uname	sigaddset
posix_spawnattr_destroy	fork	unsetenv	sigdelset
posix_spawnattr_getflags		pipe	sigemptyset
posix_spawnattr_setflags	***Timers***	confstr	sigfillset
posix_spawnattr_getgroup	timer_create		sigignore
posix_spawnattr_setgroup	timer_delete	***Memory Managemet - ANSI***	siginterrupt
posix_spawnattr_getschedparam	timer_gettime	_malloc	sigismember
posix_spawnattr_setschedparam	timer_settime	_free	sigprocmask
posix_spawnattr_getschedpolicy		shm_open	abort
posix_spawnattr_setschedpolicy	***Regular Expressions***	mlock	kill
posix_spawnattr_getsigdefault	regerror	munlock	pause
posix_spawnattr_setsigdefault	regexec	mlockall	raise
posix_spawnattr_getsigmask	regfree	munlockall	
posix_spawnattr_setsigmask	regcomp	mmap	
_exit		munmap	
atexit		msync	
execl		mprotect	

OS ABSTRACTOR®

MapuSoft Technologies, Inc.

Porting embedded applications from one OS to another OS is often an underestimated, tedious and time-consuming task. It also requires expensive and skillful resources that take away the focus on building your product. Embedded applications demand more and more performance, scalability and development flexibility from the underlying OS. Developers are forced to change their OS or extend support for more than one OS quickly as the market demands. Developers find that they need to leverage the existing software and knowledge base when migrating to next generation platforms. This has brought a need for the development of highly re-usable software that can run across proprietary and multiple commercial operating systems as well as utilizes open source components or other low cost alternatives.

It's not easy for developers to adapt existing software to a new OS or enable it to support multiple operating systems without incurring high costs and increasing time to market entry. MapuSoft offers OS PAL, OS Abstractor and OS Changer products to help developers streamline development processes and re-use their embedded software on one or more operating systems. MT offers porting, integration, support and training services to help developers easily migrate from legacy platforms to the next generation.

MapuSoft Custom Services

- Provide full porting, integration and validation services
- Extend OS Changer APIs
- Migrate in-house abstraction to OS Abstractor framework
- Add OS Abstractor support to your proprietary operating system
- Offer on-site and off-site training on operating systems and advanced porting techniques

FREE POSIX OS Abstractor Trial Software

Go to: www.mapusoft.com/downloads

➤Receive 30 days of FREE technical support!

MapuSoft Technologies, Inc.
1301 Azalea Road, Mobile, AL 36693 USA
Toll Free: 1-877-MAPUSOFT (1-877-627-8763)
Tel: 251-665-0280, Fax: 251-665-0288

www.MapuSoft.com

MAPUSOFT®
Porting Made Easy℠