POSIX IPC

	Semaphores		Shared memory	Message queues
	named	unnamed		
libraries	librt.*		librt.*	librt.*
header files	<semaphore.h>		<sys/mman.h> <sys/types.h> <fcntl.h>	<mqueue.h>
create/open	sem_open	sem_init	shm_open ftruncate mmap	mq_open
control				mq_getattr mq_setattr
IPC	sem_post sem_wait sem_getvalue			mq_send mq_receive mq_notify
close	sem_close sem_unlink	sem_destroy	munmap close shm_unlink	mq_close mq_unlink
information	sem_overview		shm_overview	mq_overview

POSIX Semaphores

	named	unnamed
libraries	librt.*	
header files	<semaphore.h>	
create/open	sem_t *sem_open(const char *name, int oflag); sem_t *sem_open(const char *name, int oflag, mode_t mode, unsigned int value);	int sem_init(sem_t *sem, int pshared, unsigned int value);
control		

IPC	int sem_post(sem_t *sem);
	int sem_wait(sem_t *sem);
	int sem_trywait(sem_t *sem);
	int sem_getvalue(sem_t *sem, int *sval);	

close	int sem_close(sem_t *sem); int sem_unlink(const char *name);	int sem_destroy(sem_t *sem);
information	sem_overview	
comment.	open(): name /name sh> ls /dev/shm/sem.name sh> unlink /dev/shm/sem.name	

	POSIX Shared memory
libraries	librt.*
header files	<sys/mman.h> <sys/types.h> <fcntl.h>
create/open	int **shm_open**(const char *name, int oflag, mode_t mode); int **ftruncate**(int fd, off_t length); void ***mmap**(void *addr, size_t len, int prot, int flags, int fildes, off_t off);
control	
IPC	
close	int **munmap**(void *start, size_t length); int **close**(int fd); int **shm_unlink**(const char *name);
information	shm_overview
comment	open(): name /name sh> ls /dev/shm/name sh> unlink /dev/shm/name

POSIX Message queues	
libraries	librt.*
header files	<mqueue.h>
create/open	mqd_t **mq_open**(const char *name, int oflag); mqd_t **mq_open**(const char *name, int oflag, mode_t mode, mq_attr attr);
control	mqd_t **mq_getattr**(mqd_t mqdes, struct mq_attr *attr); mqd_t **mq_setattr**(mqd_t mqdes, struct mq_attr *newattr, struct mq_attr *oldattr);
IPC	mqd_t **mq_send**(mqd_t mqdes, const char *msg_ptr, size_t msg_len, unsigned msg_prio); mqd_t **mq_receive**(mqd_t mqdes, char *msg_ptr, size_t msg_len, unsigned *msg_prio); mqd_t **mq_notify**(mqd_t mqdes, const struct sigevent *notification);
close	mqd_t **mq_close**(mqd_t mqdes); mqd_t **mq_unlink**(const char *name);
information	mq_overview
comment	open(): name /name sh> ls /dev/mqueue/name

Other POSIX IPC (also) for processes

	Mutexes	Condition variables	Barriers
libraries	libpthread.*	libpthread.*	libpthread.*
header files	<pthread.h>	<pthread.h>	<pthread.h>
create/open	pthread_mutex_init	pthread_cond_init	pthread_barrier_init
control	pthread_mutexattr_init pthread_mutexattr_setpshared	pthread_condattr_init pthread_condattr_setpshared	pthread_barrierattr_init pthread_barrierattr_setpshared
IPC	pthread_mutex_lock pthread_mutex_unlock	pthread_cond_signal pthread_cond_broadcast pthread_cond_wait pthread_cond_timedwait	pthread_barrier_wait
close	pthread_mutex_destroy	pthread_cond_destroy	pthread_barrier_destroy
information			

POSIX Mutexes	
libraries	libpthread.*
header files	<pthread.h>
create/open	pthread_mutex_t mutex = PTHREAD_MUTEX_INITIALIZER; int **pthread_mutex_init**(pthread_mutex_t *restrict mutex, const pthread_mutexattr_t *restrict attr);
control	int **pthread_mutexattr_init**(pthread_mutexattr_t *attr); int **pthread_mutexattr_destroy**(pthread_mutexattr_t *attr); int **pthread_mutexattr_setpshared**(pthread_mutexattr_t *attr, int pshared); int **pthread_mutexattr_getpshared**(const pthread_mutexattr_t *restrict attr, int *restrict pshared);
IPC	int **pthread_mutex_lock**(pthread_mu ex_t *mutex); int **pthread_mutex_trylock**(pthread_mutex_t *mutex); int **pthread_mutex_unlock**(pthread_mutex_t *mutex);
close	int **pthread_mutex_destroy**(pthread_mutex_t *mutex);
information	
comment	With processes, use **pthread_mutexattr_setpshared**()

POSIX Condition variables	
libraries	libpthread.*
header files	<pthread .h>
create/open	pthread_cond_t cond = PTHREAD_COND_INITIALIZER; int **pthread_cond_init**(pthread_cond_t *restrict cond, const pthread_condattr_t *restrict attr);
control	int **pthread_condattr_init**(pthread_condattr_t *attr); int **pthread_condattr_destroy**(pthread_condattr_t *attr); int **pthread_condattr_setpshared**(pthread_condattr_t *attr, int pshared); int **pthread_condattr_getpshared**(const pthread_condattr_t *restrict attr, int *restrict pshared);
IPC	int **pthread_cond_signal**(pthread_cond_t *cond); int **pthread_cond_broadcast**(pthread_cond_t *cond); int **pthread_cond_wait**(pthread_cond_t *restrict cond, pthread_mutex_t *restrict mutex); int **pthread_cond_timedwait**(pthread_cond_t *restrict cond, pthread_mutex_t *restrict mutex, const struct timespec *restrict abstime);
close	int **pthread_cond_destroy**(pthread_cond_t *cond);
information	
comment	With processes, use **pthread_condattr_setpshared**()

POSIX Barriers	
libraries	libpthread.*
header files	<pthread .h>
create/open	int **pthread_barrier_init**(pthread_barrier_t *restrict barrier, const pthread_barrierattr_t *restrict attr, unsigned count);
control	int **pthread_barrierattr_init**(pthread_barrierattr_t *attr); int **pthread_barrierattr_destroy**(pthread_barrierattr_t *attr); int **pthread_barrierattr_setpshared**(pthread_barrierattr_t *attr,int pshared); int **pthread_barrierattr_getpshared**(const pthread_barrierattr_t *restrict attr, int *restrict pshared);
IPC	int **pthread_barrier_wait**(pthread_barrier_t *barrier);
close	int **pthread_barrier_destroy**(pthread_barrier_t *barrier);
information	
comment	With processes, use **pthread_barrierattr_setpshared**()