

Ateneo de Manila University

Threads



Department of Information Systems and
Computer Science

S.Y. 2001-2002

<http://sysads.ateneo.net/wyu/>

wyy@admu.edu.ph

POSIX Threads

- ★ POSIX standardized implementation for threads
- ★ are simply lightweight processes (implement using `clone ()`)
- ★ shared with the parent process:
 - Address Space
 - File Descriptor Table
 - Signal Handling Context
- ★ defined in the `pthread.h` header file
- ★ methods found in the `libpthread` library

Benefits of Threads

- ★ less time to create a new thread than a process
- ★ less time to terminate a thread than a process
- ★ less time to switch between two threads within the same process
- ★ less communication overheads (use address space)

POSIX Threads: Essentials

```
★ int pthread_create(pthread_t * thread,  
pthread_attr_t *attr, void *  
(*start_routine)(void *), void * arg);
```

- creates a new thread that shall execute the `start_routine()`
- the argument of the `start_routine()` is passed in `arg`
- the `start_routine()` must either return or terminate with the `pthread_exit()` function
- `attr` can be set to `NULL` for default thread attributes which are joinable and non real-time scheduling policy
- returns a non-zero value if an error occurs

```
★ int pthread_join(pthread_t th, void  
**thread_return);
```

- suspends the execution of the calling thread until the thread identified by `th` terminates
- the return value of `th` can be retrieved from `thread_return`
- must be called for unused joinable thread to prevent a memory leak
- returns a non-zero on error

★ `int pthread_detach(pthread_t th);`

- this function makes thread `th` detached (this means it's memory space will be deallocate when the thread terminates)
- returns a non-zero on error

★ `void pthread_exit(void *retval);`

- terminates a thread
- returns a non-zero on error

```
★ pthread_t pthread_self(void);
```

- returns the current thread id of the current thread
- returns a -1 on error

```
★ int pthread_equal(pthread_t tid1, pthread_t  
tid2);
```

- determines if two thread identifiers refer to the same thread
- returns a -1 on error

POSIX Threads: Setting Attributes

- ★ `int pthread_attr_init(pthread_attr_t *attr);`
- ★ `int pthread_attr_destroy(pthread_attr_t *attr);`
 - create or destroy a `pthread_attr` structure which can be passed to `pthread_create()`
 - returns -1 if an error occurs
- ★ `int pthread_attr_setXXXXX(pthread_attr_t *attr, int XXXXX);`
- ★ `int pthread_attr_getXXXXX(const pthread_attr_t *attr, int *XXXXX);`
 - set or get the value of `XXXXX` from the `pthread_attr` structure
 - returns -1 if an error occurs
 - values for `XXXXX` are:
 - * `detachstate` - sets/gets whether a thread is joinable

- * schedpolicy - sets/gets the scheduling policy
 - * schedparam - sets/gets scheduling parameters
 - * inheritsched - sets/gets whether new threads will follow scheduling policy and parameters of this thread
 - * scope - set/gets what the thread contend for (typically CPU)
- returns a non-zero on error

POSIX Threads: Setting Thread Specific Data

- ★ `int pthread_key_create(pthread_key_t *key, void (*destr_function) (void *));`
- ★ `int pthread_key_delete(pthread_key_t key);`
- ★ `int pthread_setspecific(pthread_key_t key, const void *pointer);`
- ★ `void * pthread_getspecific(pthread_key_t key);`
 - manipulate thread-specific data for threads
 - `pthread_key_create ()` creates the thread specific data which is identified by `key` and an optional pointer to a destructor function
 - `pthread_key_delete ()` removes the thread specific data which is identified by `key`
 - `pthread_setspecific ()` assigns the data pointed by `pointer` to the memory region defined by `key`

- `pthread_getspecific()` retrieves the data pointed by pointer to the memory region defined by `key`

POSIX Threads: Scheduling

```
★ int pthread_once(pthread_once_t *once_control,  
void (*init_routine)(void));
```

- initialization of thread that is executed only once
- this function ensures that the `init_routine()` is executed at least once
- returns a -1 on error

```
★ int sched_yield(void)
```

- defined in the `sched.h` header file
- current thread yields execution in favor of another thread
- process is moved to end of the process queue
- returns a -1 on error

```
* int pthread_setschedparam(pthread_t  
target_thread, int policy, const struct  
sched_param *param);
```

```
* int pthread_getschedparam(pthread_t  
target_thread, int *policy, struct sched_param  
*param);
```

- sets/gets the scheduling parameters for the thread `target_thread` as indicated by `policy` and `param`
- value of `policy` can be:
 - * `SCHED_OTHER` regular and non-realtime scheduling
 - * `SCHED_RR` realtime and round-robin scheduling
 - * `SCHED_FIFO` realtime and first-in first-out scheduling
- returns a non-zero value on error

POSIX Threads: Sending Signals

```
★ int pthread_kill(pthread_t thread, int signo);
```

```
★ int pthread_sigmask(int how, const sigset_t  
  *newmask, sigset_t *oldmask);
```

```
★ int sigwait(const sigset_t *set, int *sig);
```

- `pthread_kill()` sends a signal to a particular thread
- the semantics of this command are similar to `kill()`
- `pthread_sigmask()` changes the signal mask for the calling thread based on `newmask` and `how`
- these signal masks are defined in `sigprocmask(2)`
- functions return non-zero values if an error occurs



Copyright © 2000-2001 by William Emmanuel S. Yu. This material may be distributed only subject to the terms and conditions set forth in the Open Content License, v1.0 or later (the latest version is presently available at <http://opencontent.org/opl.shtml>).