This page last changed on 2010/06/01 by hkpco.

Using Thread Pool Library

라이브러리 정보

- 라이브러리 위치: libvos/include/threadpool/*
- 라이브러리 파일: threadpool.h threadpool_queue.h threadpool.c threadpool_queue.c
- include는 "threadpool.h"만 하면 됨

threadpool.h - 구조체

```
/* threadpool에 필요한 모든 정보를 포함하는 구조체 */
typedef struct threadpool_all {
        unsigned int worker_num;
        unsigned int max_queue_size;
        unsigned int curr_queue_size;
        struct threadpool_work *work_queue_head;
        struct threadpool_work *work_queue_tail;
        pthread_t *threads;
        pthread_mutex_t work_lock;
        pthread_cond_t work_pool_available;
        pthread_cond_t work_pool_exist;
        pthread_cond_t work_empty;
        int working;
} thpool;
```

- 설명
  threadpool에 필요한 정보들을 구조체 형식으로 묶어두었다.
  worker_num과 max_queue_size는 각각 생성할 스레드 풀의 개수와 작업을 할당할 work pool의 최대 사이즈이다.
  curr_queue_size는 현재 work_pool에 할당된 작업의 개수를 나타낸다.
  work_queue_head, work_queue_tail은 work pool을 위한 queue의 head와 tail을 나타낸다.
  threads는 pthread_create() 함수에 의해 생성된 스레드의 ID값이 저장된다.
  work_lock은 스레드 풀의 작업 수행 함수에서 전역적으로 사용되는 데이터들에 대한 mutex lock을 위해 선언되었다.
  work_pool_available은 work pool에 할당할 수 있는 작업 공간이 있다는 것에 대한 신호를 보내기 위해 선언되었다.
  work_pool_exist는 작업이 할당된 뒤, 작업 수행 함수에게 현재 할당된 작업이 있다는 신호를 보내기 위해 선언되었다.

threadpool.h - 함수

```
int threadpool_init( thpool **init_pool, unsigned int worker_num, unsigned int max_queue );
```

- threadpool의 초기화를 위한 함수이다.
- 첫 번째 인자인 init_pool은 thpool 구조체 변수가 전달되며, 해당 변수에 대한 별도의 작업은 필요하지 않다.
- 두 번째 인자인 worker_num은 thread pool을 위한 스레드의 개수가 전달된다.
- 세 번째 인자인 max_queue는 work queue의 크기를 나타낸다.

```
void *threadpool_worker( void *arg );
```

- queue에 할당된 작업을 수행하기 위한 함수로, 라이브러리 내부에서 실행되기 때문에 사용자는 고려할 필요가 없다.

```
int threadpool_add_work( thpool *th_pool, void *(*worker)(void *), void *worker_arg );
```

- thread pool에 작업을 할당하는 함수이다.
- 첫 번째 인자인 th_pool에는 이전에 초기화 함수를 통하여 전달되었던 첫 번째 변수가 그대로 전달된다.
- 두 번째 인자인 worker은 실제 스레드가 수행할 함수가 전달된다.
- 세 번째 인자인 worker_arg는 그에 대한(스레드 함수) 인자가 전달된다.

```
int threadpool_adjust( thpool **adj_pool, unsigned int worker_num, unsigned int max_queue );
```

- 중간에 thread pool 및 work pool의 크기를 조절하기 위한 함수로써, 현재는 threadpool의 destroy&init를 wrapping하는것으로 대체되었다.
- 첫 번째 인인 adj_pool은 이전에 초기화 함수를 통하여 전달되었던 첫 번째 변수가 그대로 전달된다.
- 두 번째와 세 번째 인자인 worker_num과 max_queue는 각각 thread pool의 크기, work pool의 크기이다.

```
int threadpool_destroy( thpool *thp );
```

- thread pool의 종료 시 소멸 과정을 수행하는 함수이다.
- 인자인 thp에는 이전에 초기화 함수를 통하여 전달되었던 첫 번째 변수가 그대로 전달된다.

```
static int __inline__ work_queue_none( thpool *thp )
{
	return (thp->curr_queue_size == 0);
}
```

- 현재 work pool이 비어있는지 체크하는 함수
- thread pool 라이브러리 내부에서 호출되므로 사용자는 해당 함수를 고려할 필요가 없다.

```
static int __inline__ work_queue_full( thpool *thp )
{
	return (thp->curr_queue_size == thp->max_queue_size);
}
```

- 현재 work pool이 최대치인지 체크하는 함수
- thread pool 라이브러리 내부에서 호출되므로 사용자는 해당 함수를 고려할 필요가 없다.

```
/* 에러 핸들링 함수, 에러 처리는 모두 해당 함수로 했기 때문에,
 * 차후 함수를 수정하여 콘솔 출력 등으로 바꿔 사용 할 수 있음. */
static void __inline__ thp_error_handling( char *str )
{
	char err_msg[512];

	snprintf( err_msg, sizeof(err_msg) -1, "threadpool> %s", str );
	perror(err_msg);
}
```

- thread pool 내부에서 사용되는 error handling 전용 함수이다.
- 에러 처리를 위한 루틴은 따로 wrapping해 두었으며, 수정하여 사용이 가능하다.
- thread pool 라이브러리 내부에서 호출되므로 사용자는 해당 함수를 고려할 필요가 없다.

threadpool_queue.h - 구조체

```
/* queue 형태로 되어있는 work 구조체 */
typedef struct threadpool_work {
        void *(*worker_func)(void *);
        void *worker_arg;
        struct threadpool_work *prev;
        struct threadpool_work *next;
} thpwork;
```

- 설명
  스레드가 수행할 작업을 work queue 형태로 할당하기 위해 만들어진 구조체로,
  worker_func와 worker_arg는 각각 수행할 스레드 함수와 그에 대한 인자를 나타낸다.
  prev, next는 owrker queue의 전/후를 가리키는 linked list이다.

threadpool_queue.h - 함수

```
void queue_init( thpwork **head, thpwork **tail );
```

- work pool을 위한 queue를 초기화하는 함수이다.
- thread pool 라이브러리 내부에서 호출되므로 사용자는 해당 함수를 고려할 필요가 없다.

```
void queue_push( thpwork *v, thpwork *head, thpwork *tail );
```

- work pool에 작업을 할당하기 위한 queue의 push 함수이다.
- thread pool 라이브러리 내부에서 호출되므로 사용자는 해당 함수를 고려할 필요가 없다.

```
thpwork *queue_pop( thpwork *head, thpwork *tail );
```

- work pool에 할당된 작업을 수행하기 위해 가져오는 queue의 pop 함수이다.
- thread pool 라이브러리 내부에서 호출되므로 사용자는 해당 함수를 고려할 필요가 없다.

```
void queue_destroy( thpwork *head, thpwork *tail );
```

- work pool의 소멸과정을 수행하는 queue의 destroy 함수이다.
- thread pool 라이브러리 내부에서 호출되므로 사용자는 해당 함수를 고려할 필요가 없다.

사용 예 - Pseudo Code

- 해당 라이브러리를 사용하면 몇 번의 함수 호출로 스레드 풀을 간단히 적용할 수 있다.

```
void *thrd_func( int *client_sock );
```

```
int main( void )
{
	thpool *th_pool; // 스레드 풀에 사용되는 변수

	...

	threadpool_init( &th_pool, 7, 64 ); // 스레드 풀에 사용되는 초기화 함수

	while(1)
	{
		cli_sfd = accept(..);

		threadpool_add_work( th_pool, (void *)thrd_func, &cli_sfd ); // 스레드 풀에 사용되는 작업 추가 함수

		if( somecheck ) {
			threadpool_adjust( &th_pool, 5, 32 ); // 스레드 풀에 사용되는 풀 정보 조절 함수
		}
	}
	threadpool_destroy( th_pool ); // 스레드 풀에 사용되는 소멸 함수

	return 0;
}
```