Computer Architecture

Discussion 6

Minye Wu
wumy@shanghaitech.edu.cn

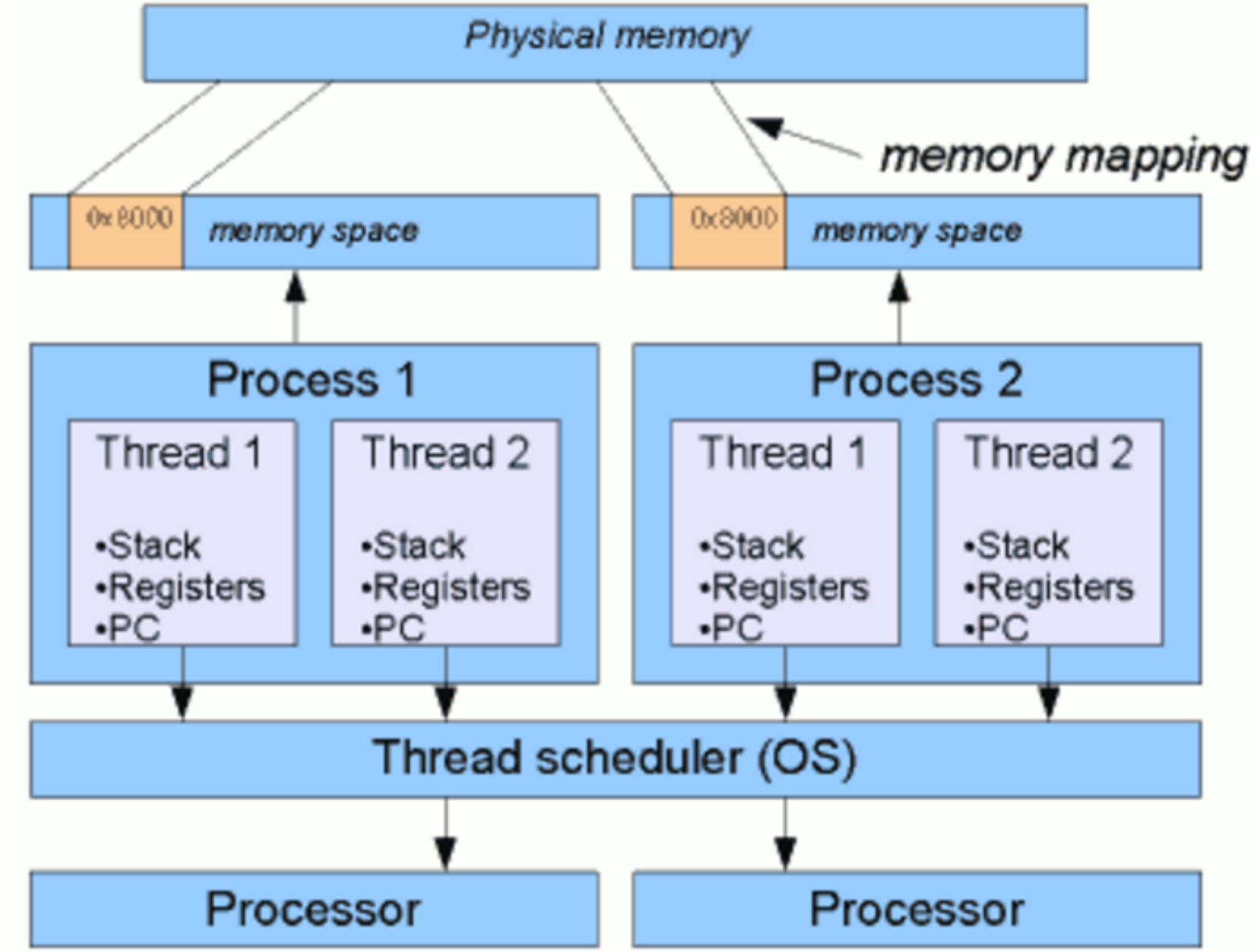
Mar. 29 2016

Keywords

- Thread
- Mutex
- Signal
- Function pointer

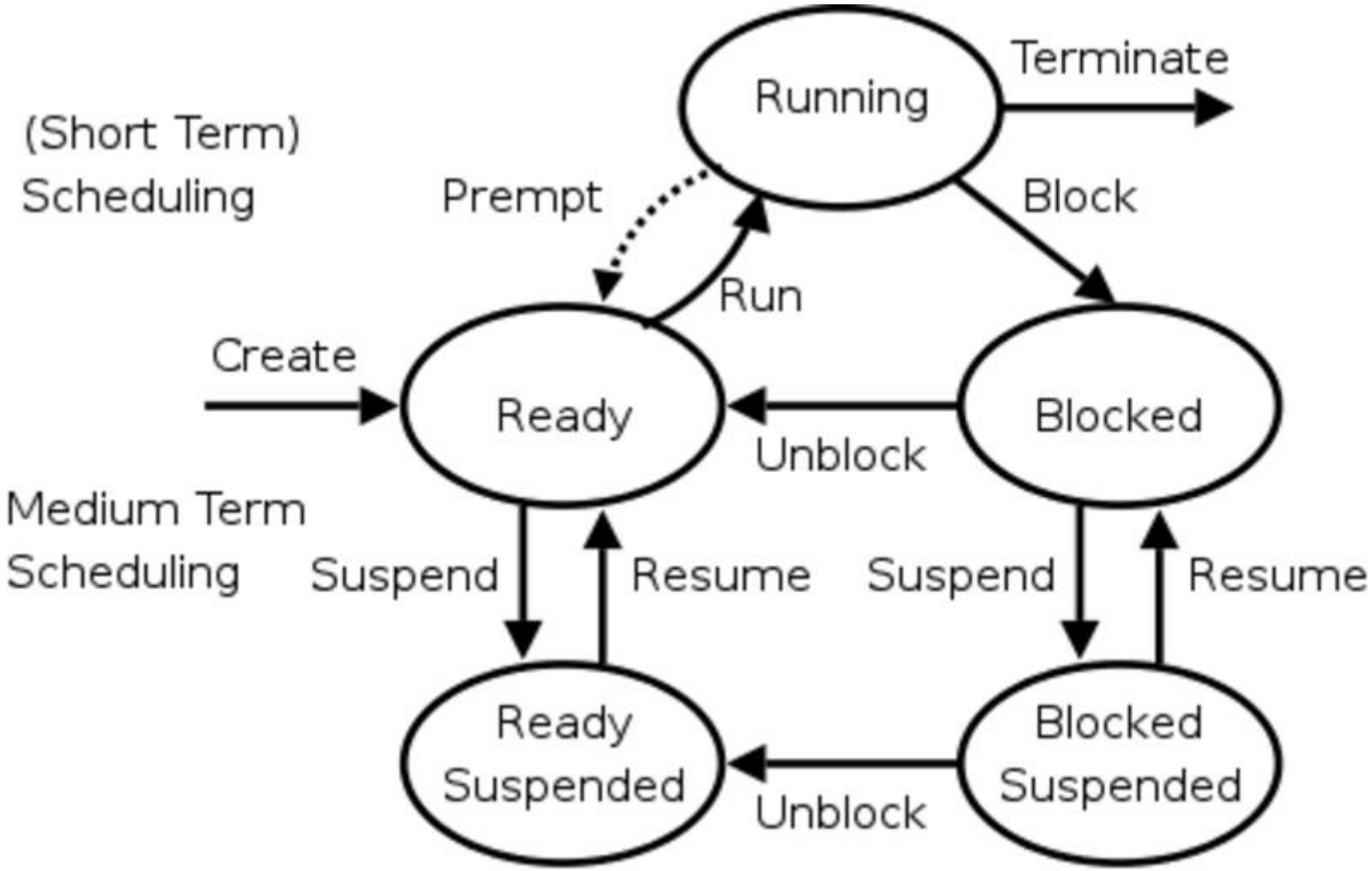
Thread

- Threads share process state such as memory, open files, etc.
- Each thread has a separate stack for procedure calls (in shared memory)
- Thread is unit of sequential execution



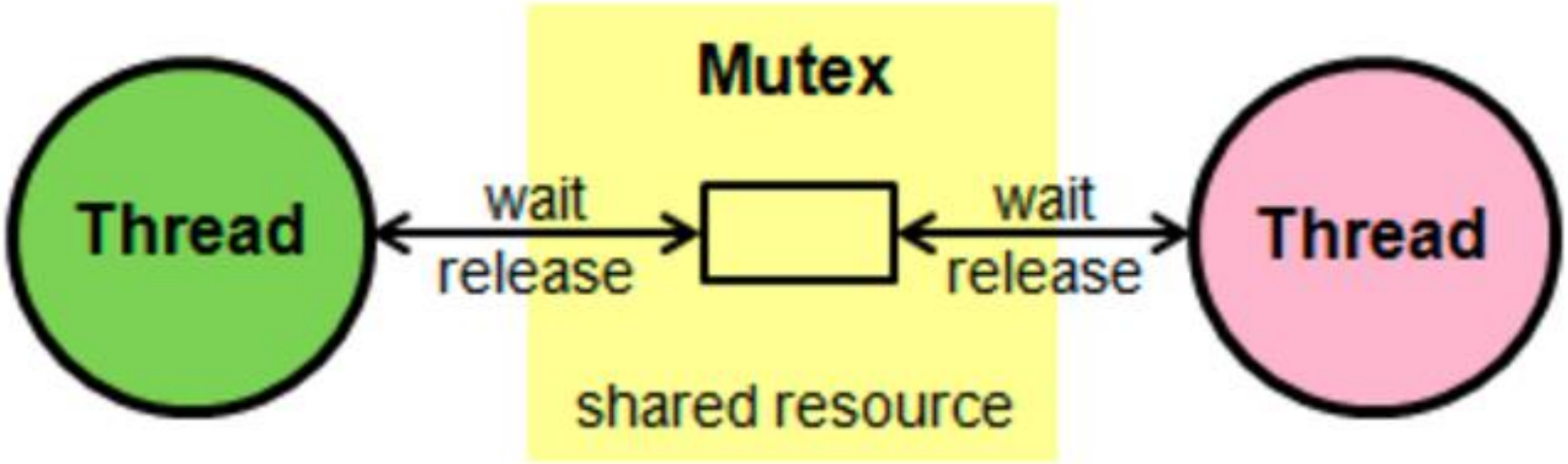
Google image

scheduling



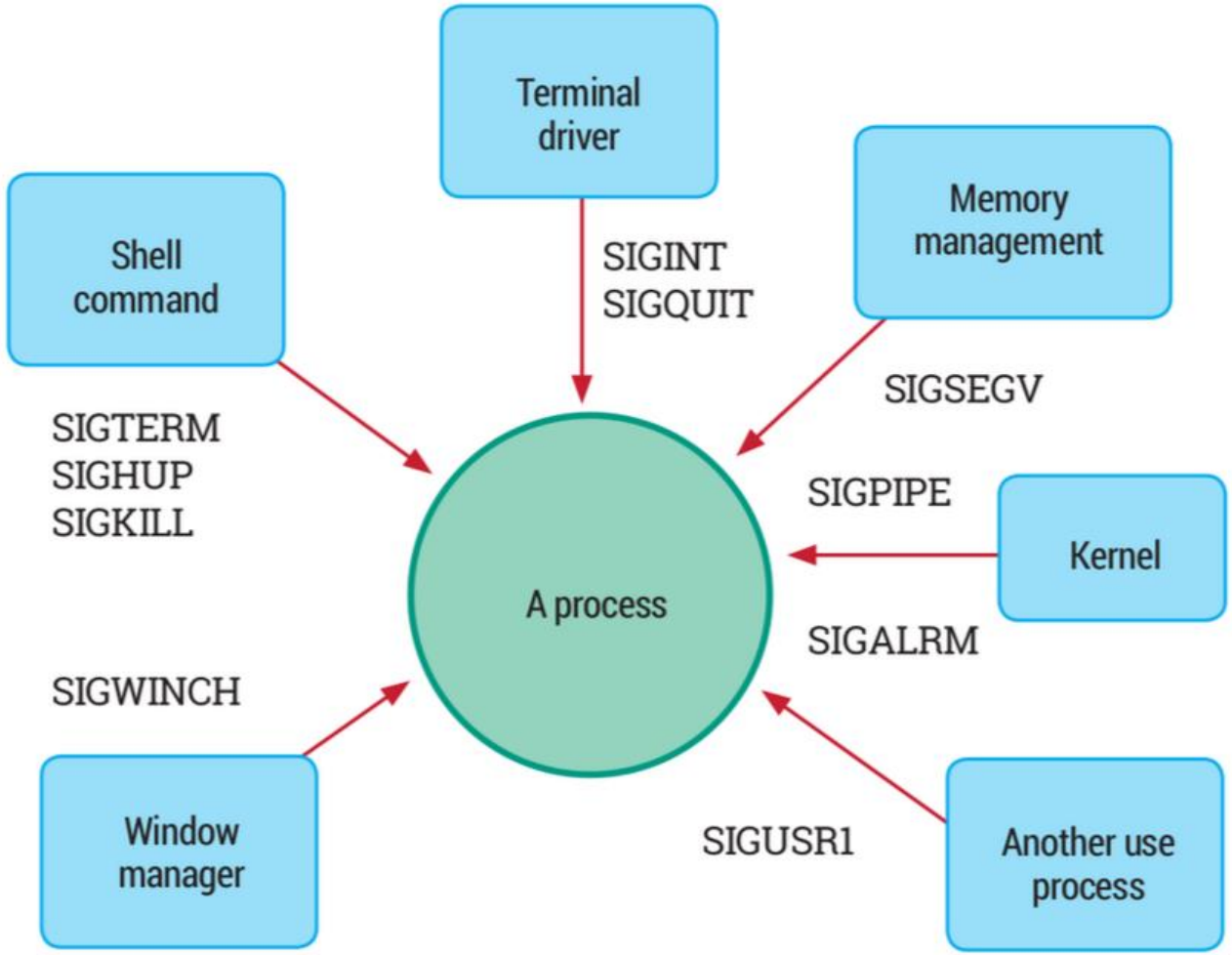
Google image

Mutex



Google image

Signal



Google image

Exercises

- #include <pthread.h>
- #include <signal.h>

- pthread_t
- int pthread_create(pthread_t *tidp,const pthread_attr_t *attr,(void*)(*start_rtn)(void*),void *arg);
- int pthread_join(pthread_t thread, void **retval);

- signal(int signo, void(*func)(int))

Mutex

- pthread_mutex_t
- int pthread_mutex_init(pthread_mutex_t *restrict mutex,const pthread_mutexattr_t *restrict attr);

- int pthread_mutex_lock(pthread_mutex_t *mutex);
- int pthread_mutex_unlock(pthread_mutex_t *mutex);

- pthread_mutex_destroy(pthread_mutex_t *mutex);

Function pointer

- Declare a function pointer as though you were declaring a function, except with a name like *foo instead of just foo:
- void (*foo)(int);

You can get the address of a function simply by naming it:

```
void foo();
func_pointer = foo;
```

or by prefixing the name of the function with an ampersand:

```
void foo();
func_pointer = &foo;
```

Function pointer

Invoke the function pointed to just as if you were calling a function.

```
func_pointer( arg1, arg2 );
```

or you may optionally dereference the function pointer before calling the function it points to:

```
(*func_pointer)( arg1, arg2 );
```