POSIX Threads programming (pthread)

Operating System

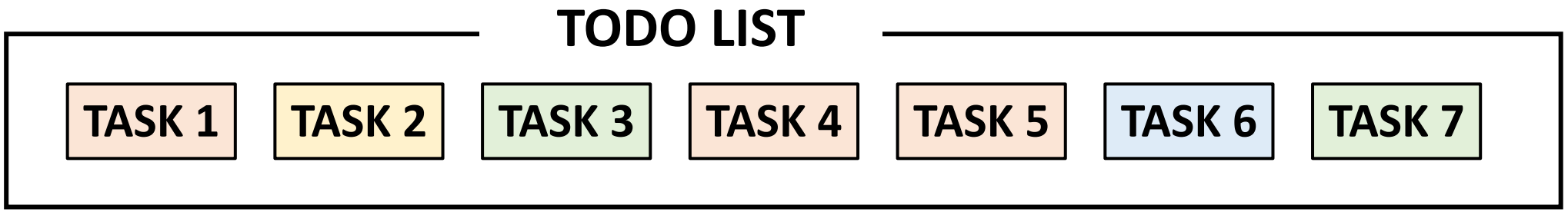
Introduction

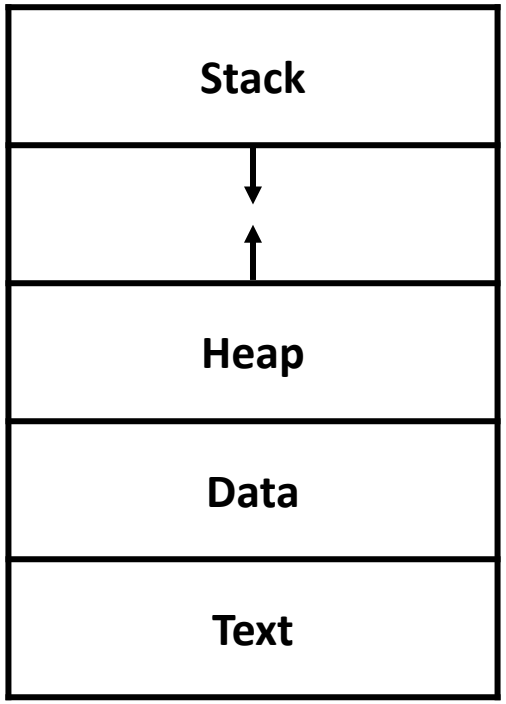
- What is a thread?
- POSIX Thread(Pthread) Overview
- The Pthread API
- Compile pthread program
- Thread Management
- Mutex variables
- Condition variables

What is a thread?

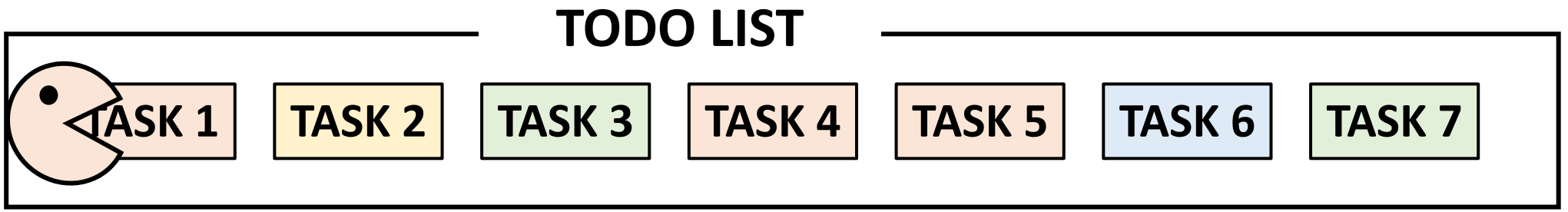
- A thread of execution is **the smallest sequence of programmed instructions** that can be managed **independently** by a scheduler, which is typically a part of the operating system.
- Threads exist **within a process** as **subsets of a process.**
- Threads may share and use the process resources.
- Thread has its **own independent flow of control** as long as its parent process exists and the OS supports it
- Thread **duplicates only the essential resources** it needs to be independently schedulable
- Thread is "**lightweight**" because most of the overhead has already been accomplished through the creation of its process.

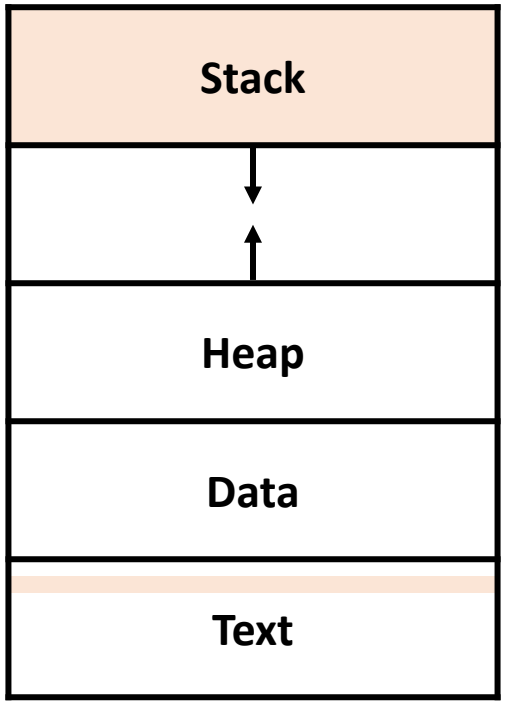
Process model



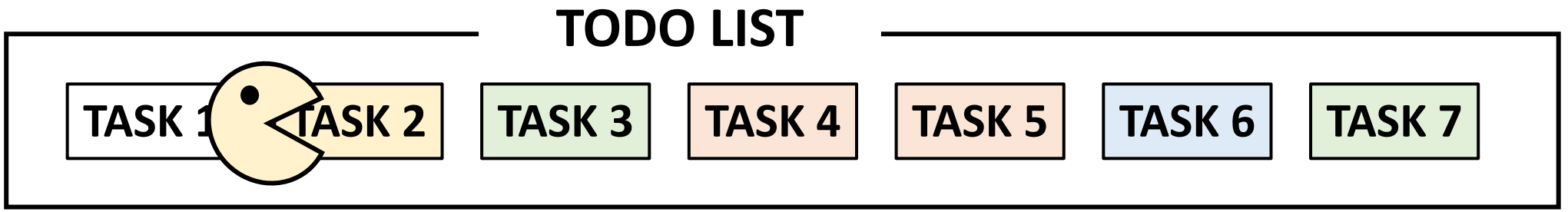


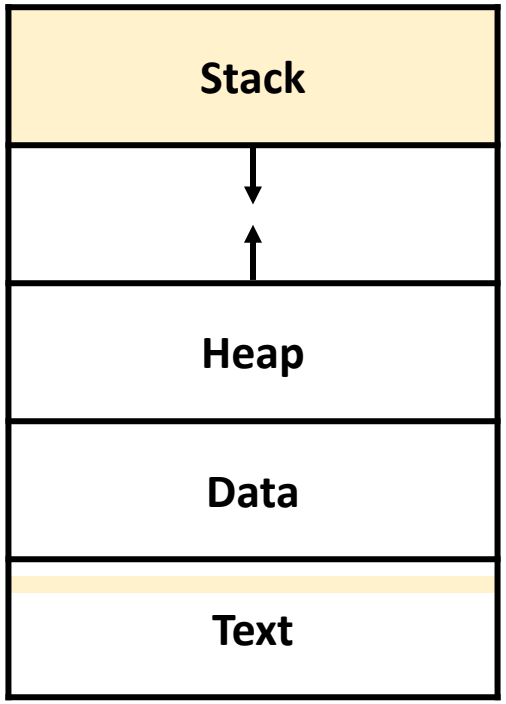
Process model



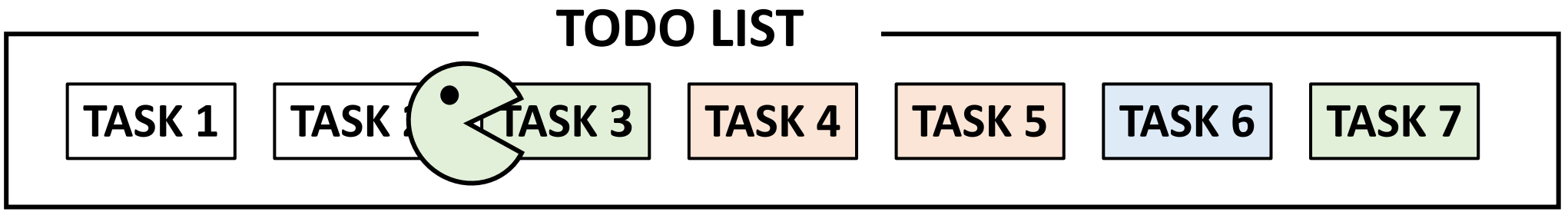


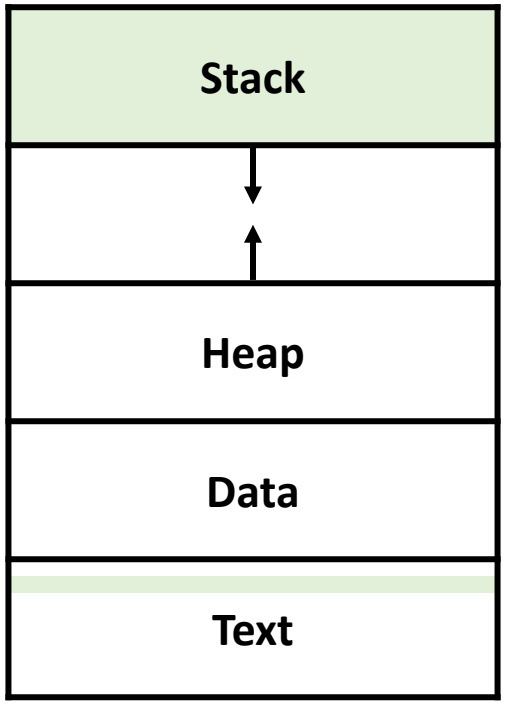
Process model



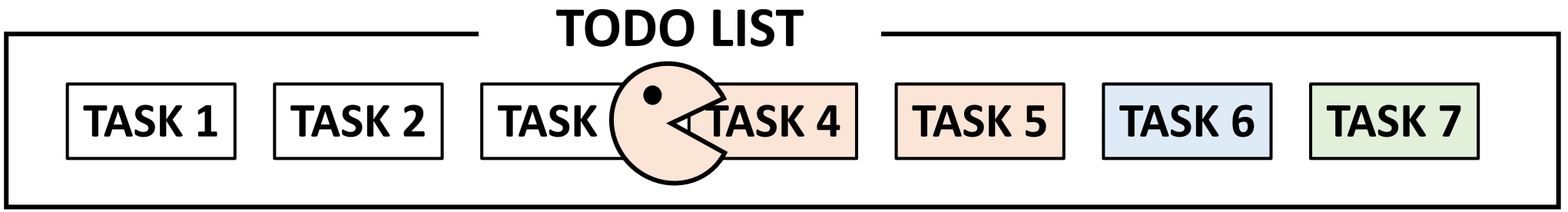


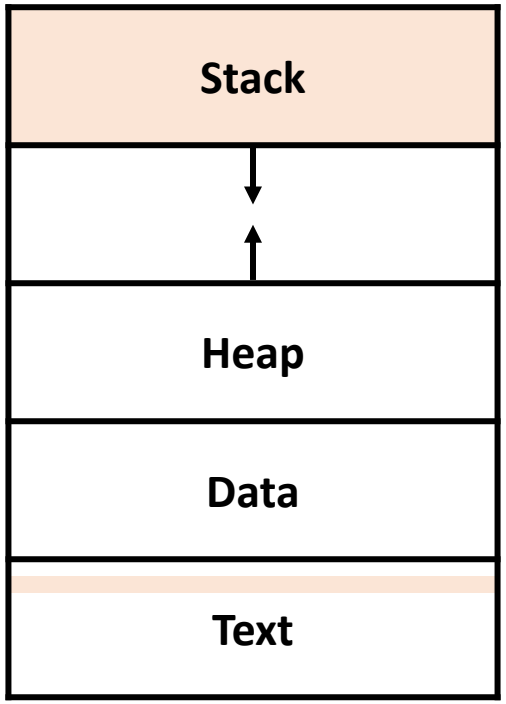
Process model



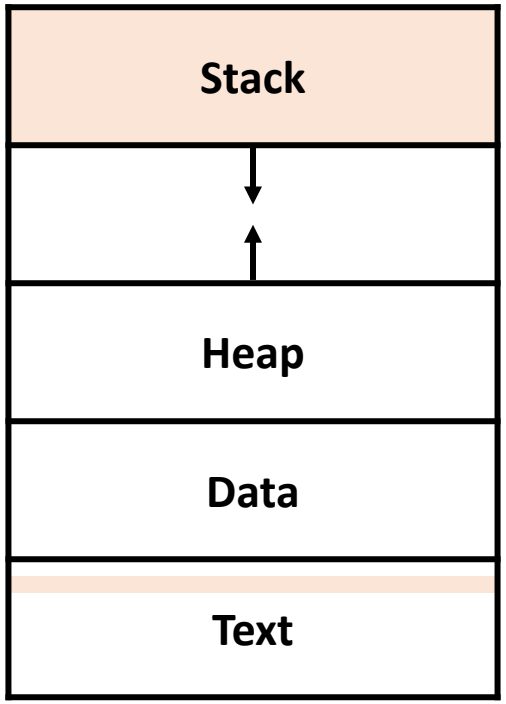


Process model

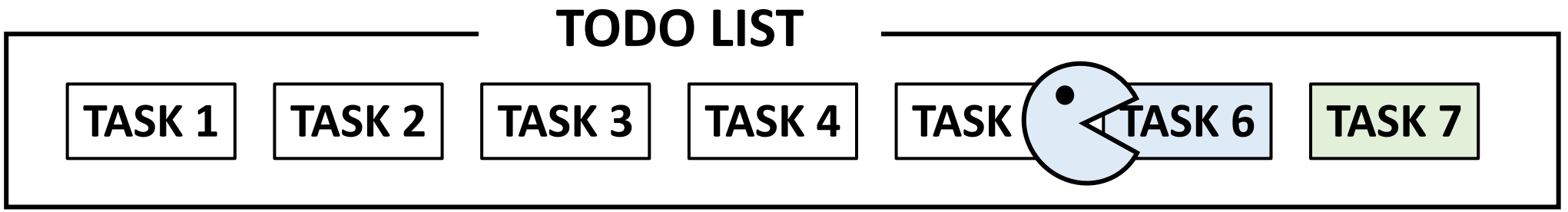


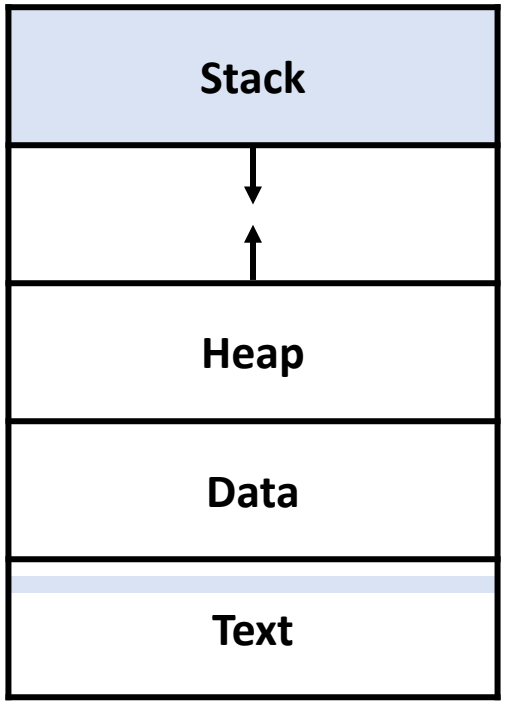


Process model

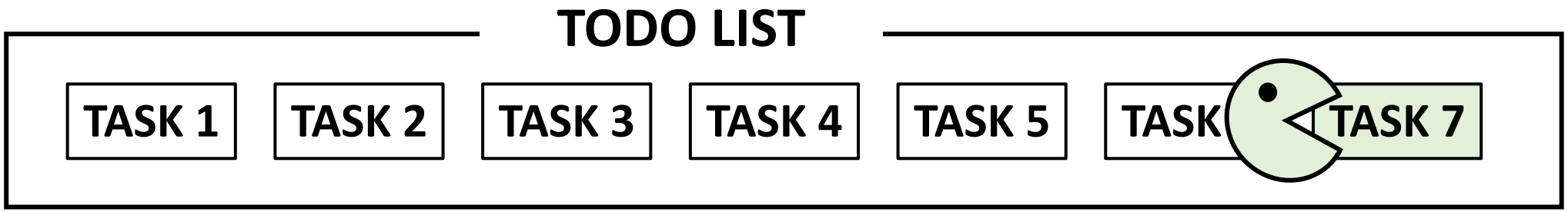


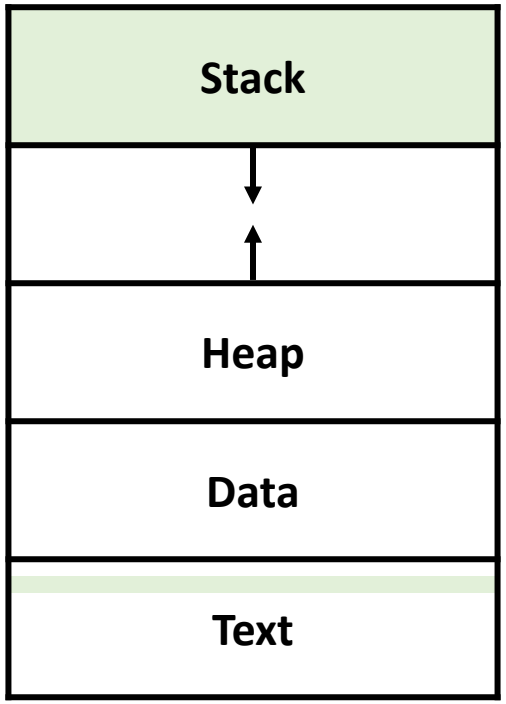
Process model



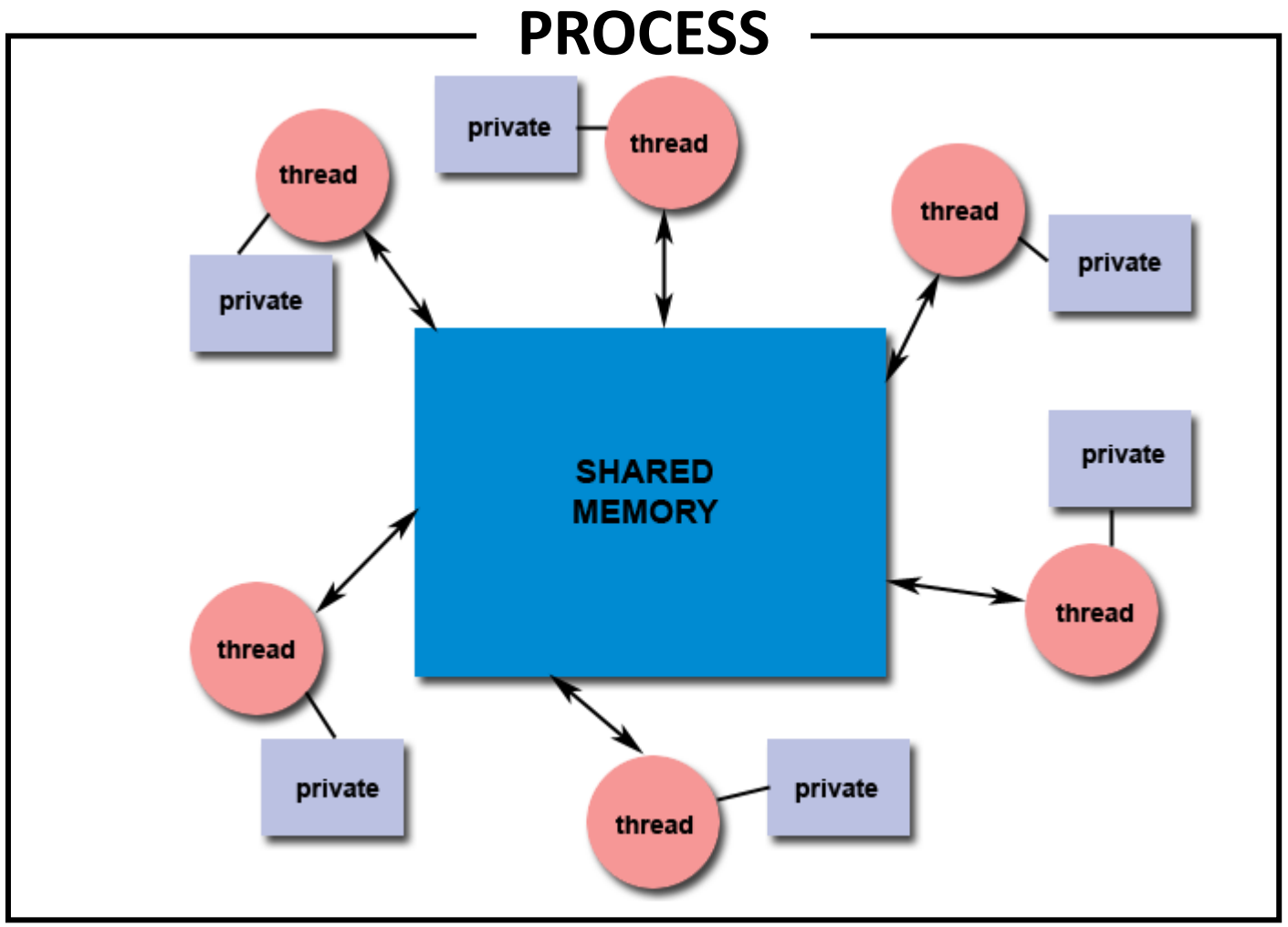


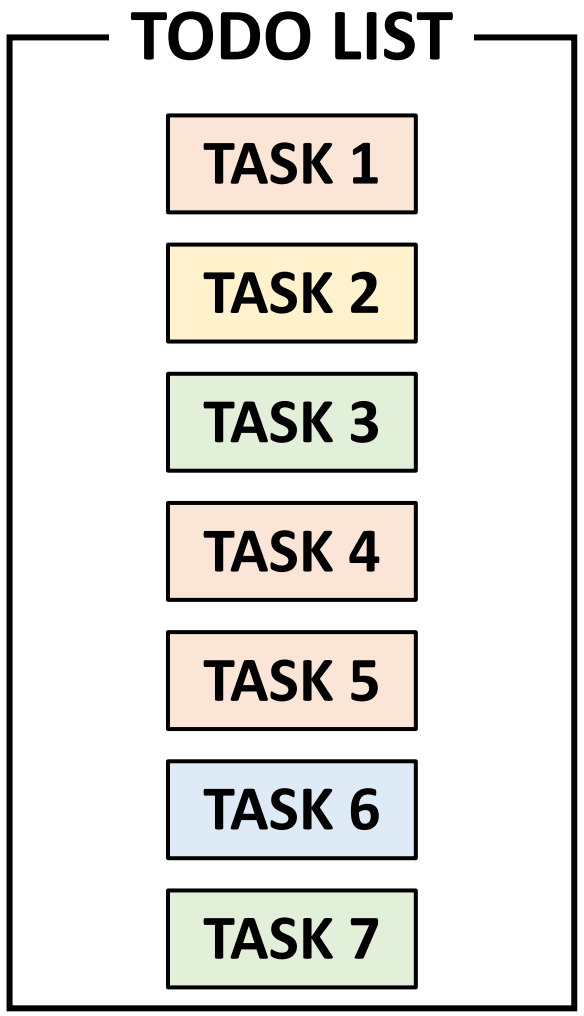
Process model



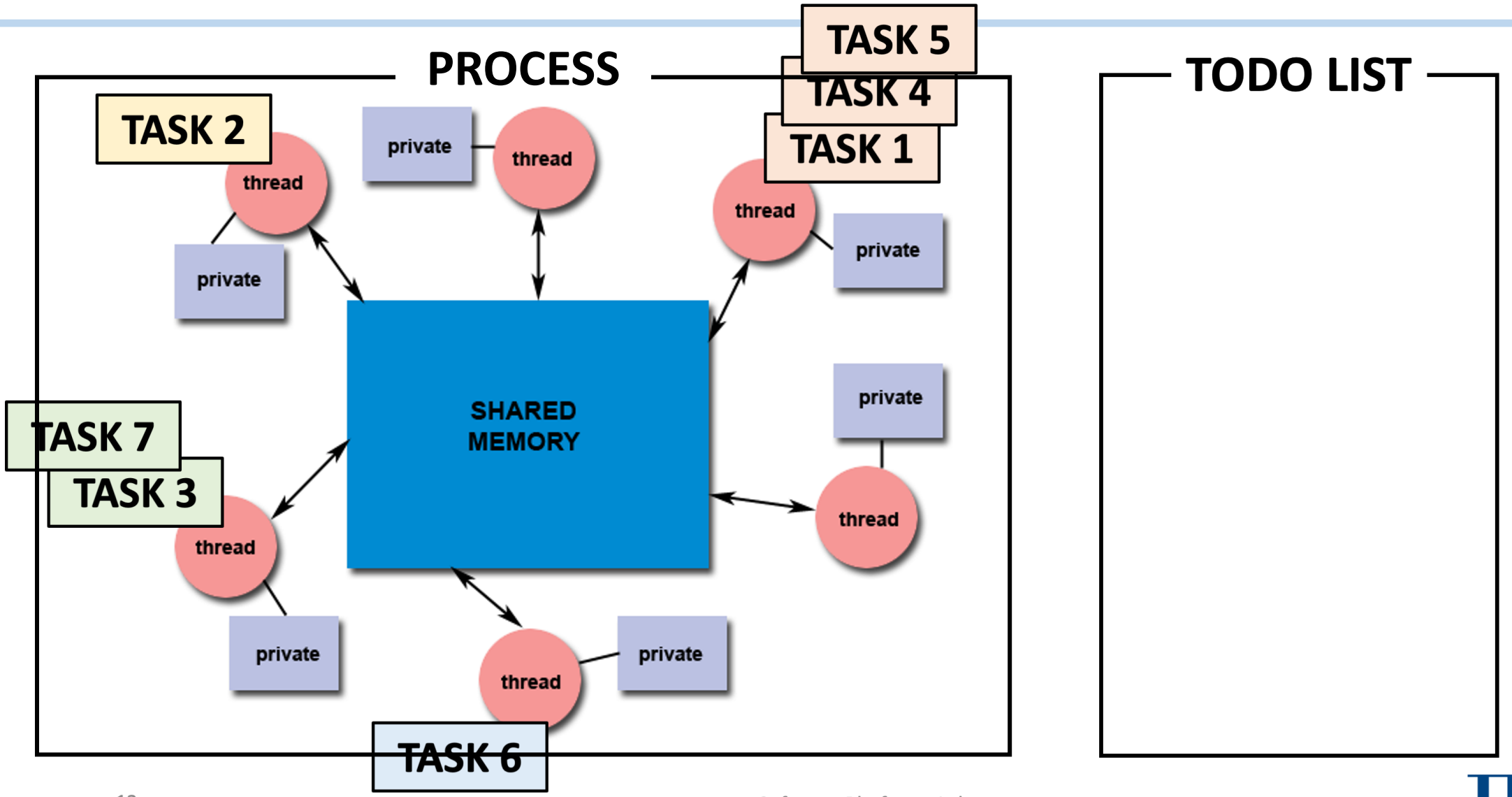


Thread model

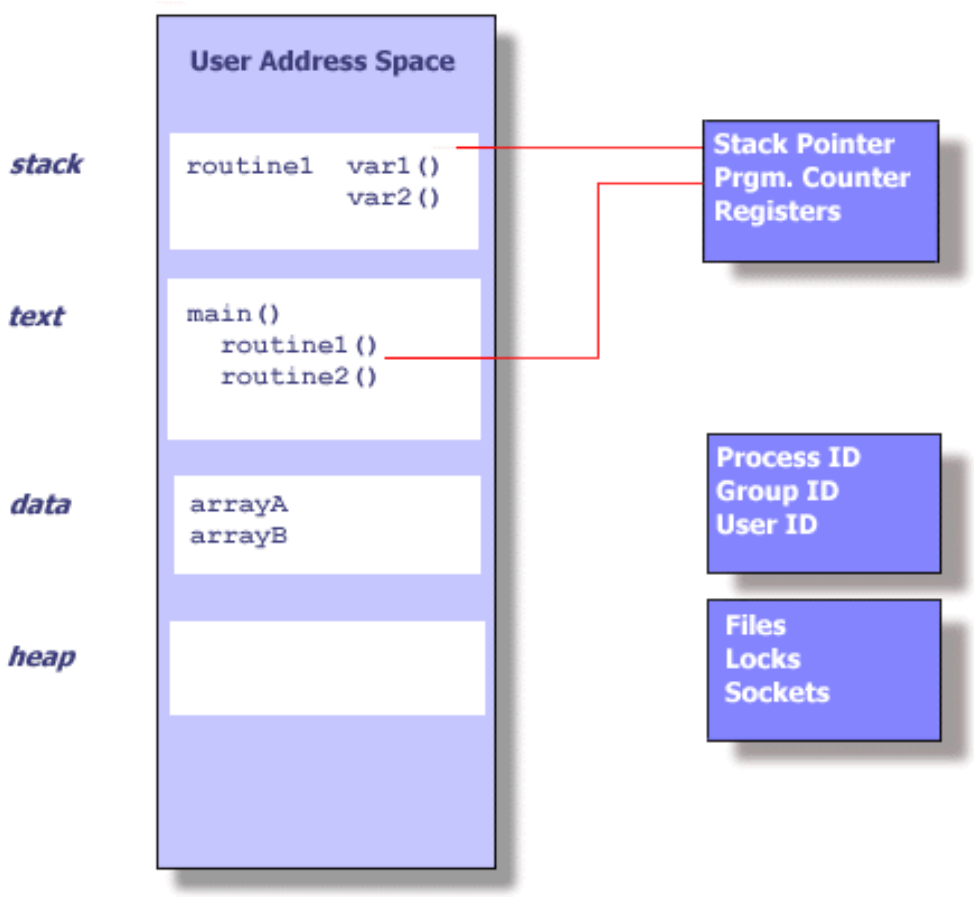




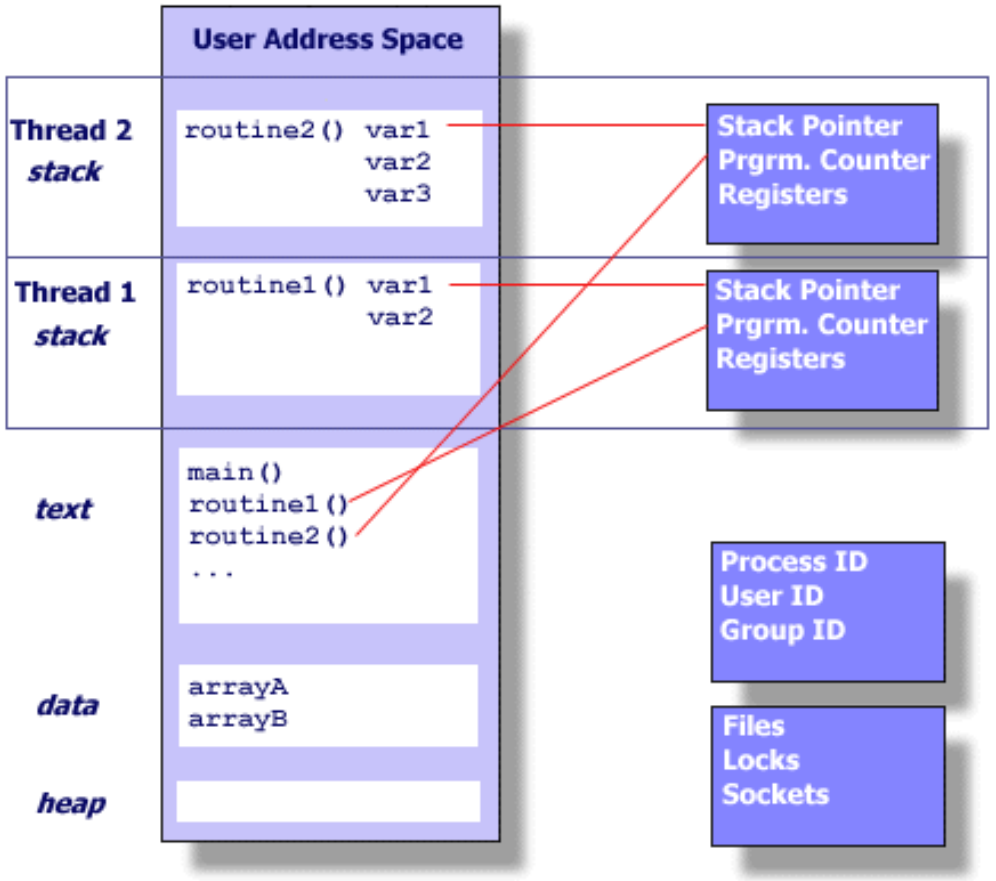
Thread model



Process vs. Thread

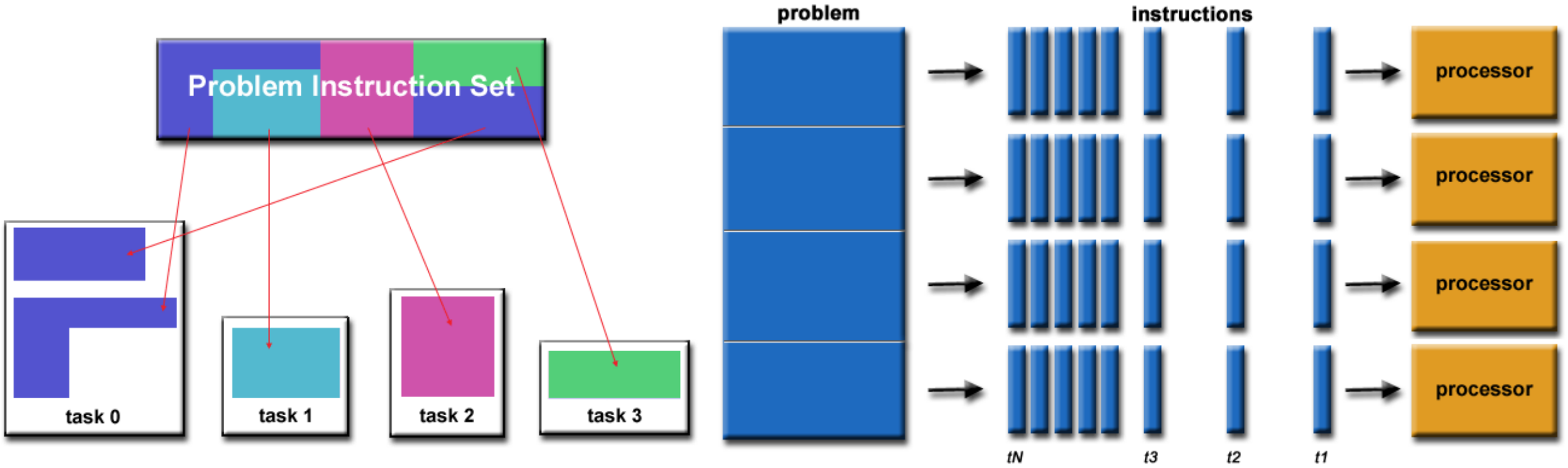


UNIX PROCESS

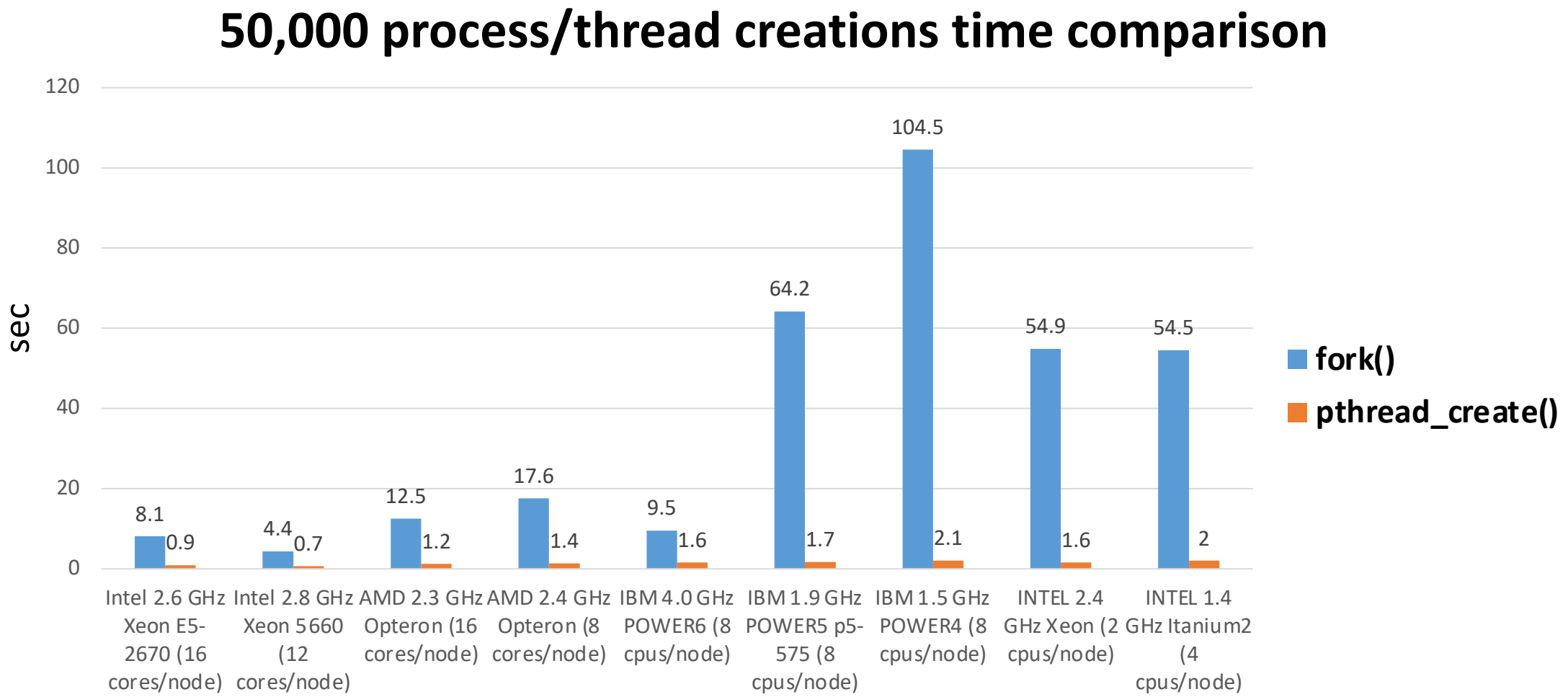


THREADS WITHIN A UNIX PROCESS

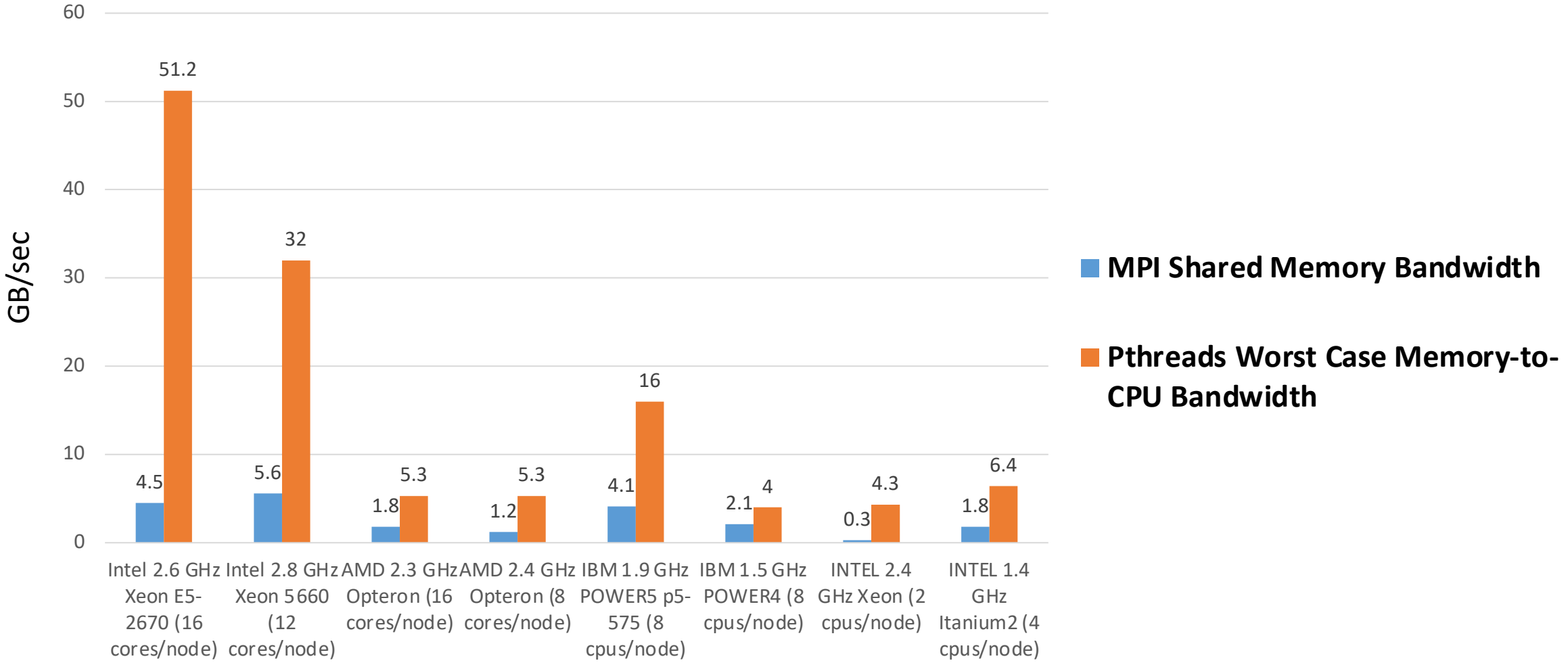
1. Parallel Programming



2. Light Weight



3. Efficient Communications/Data Exchange



The Pthread API

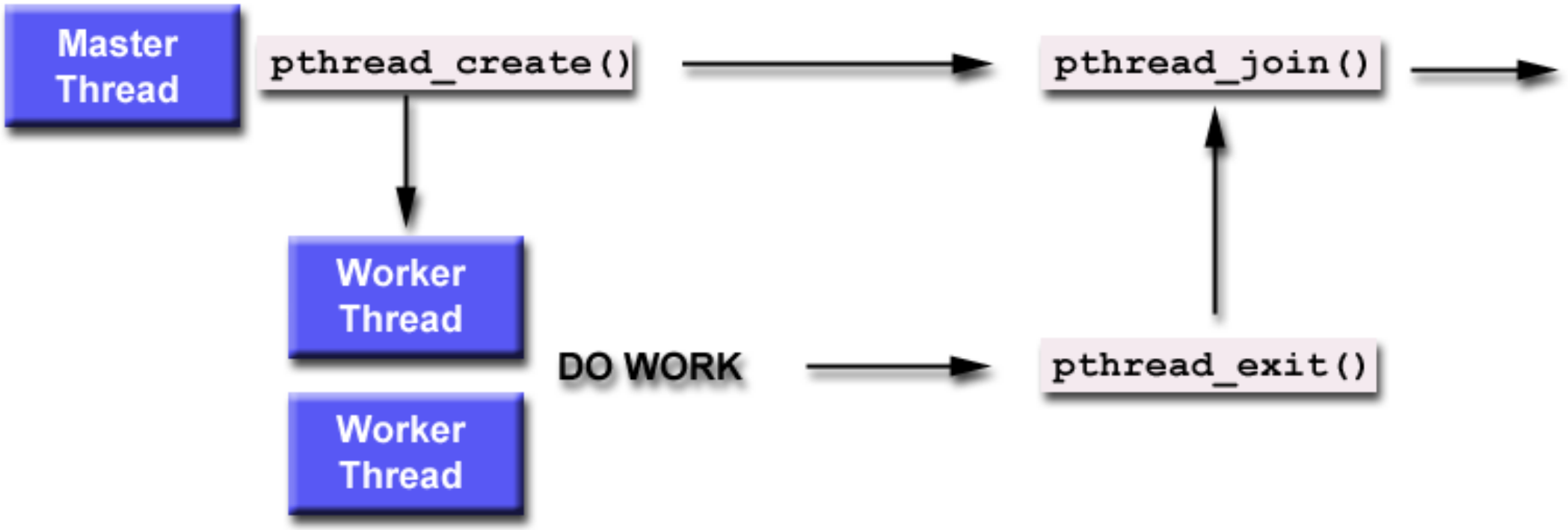
1. **Thread management:** Routines that work directly on threads - creating, detaching, joining, etc.
2. **Mutexes:** Routines that deal with synchronization, called a "mutex", which is an abbreviation for "**mut**ual **ex**clusion". Mutex functions provide for creating, destroying, locking and unlocking mutexes.
3. **Condition variables:** Routines that address communications between threads that share a mutex. Based upon programmer specified conditions. This group includes functions to create, destroy, wait and signal based upon specified variable values.

Compile pthread program

- For portability, the **pthread.h header file** should be included in each source file using the Pthreads library.

Compiler / Platform	Compiler Command	Description
INTEL Linux	icc -pthread	C
	icpc -pthread	C++
PGI Linux	pgcc -lpthread	C
	pgCC -lpthread	C++
GNU Linux, Blue Gene	gcc -pthread	GNU C
	g++ -pthread	GNU C++
IBM Blue Gene	bgxlc_r / bgcc_r	C (ANSI / non-ANSI)
	bgxlC_r, bgxlc++_r	C++

Thread Management



Pthread API – pthread_create

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

- Create a new thread.

@param [out]	thread_t	ID of new thread
@param [in]	attr	Thread attribute configuration (e.g.,Stack size)
@param[in]	start_routine	Thread function to execute (main function of new thread)
@param[in]	arg	argument(s) pass(es) to start_routine
@return		If success 0, else error num

Pthread API – pthread_join

```
int pthread_join(pthread_t thread,
                 void **ret_val);
```

- Waits for the termination of a specific thread.
  - Return immediately if the target thread already terminated.

@param [in]	thread_t	Thread ID of target thread which would terminate
@param [out]	ret_val	Return value of terminated thread
@return		If success 0

Pthread API – pthread_exit

```
void pthread_exit(void *ret_val);
```

- Terminates the thread.
  - Return call in thread start function is implicit call of pthread_exit().

@param [in] ret_val	Return value of terminated thread. Another thread which waits for the thread by pthread_join can get this value.

Mutex variables

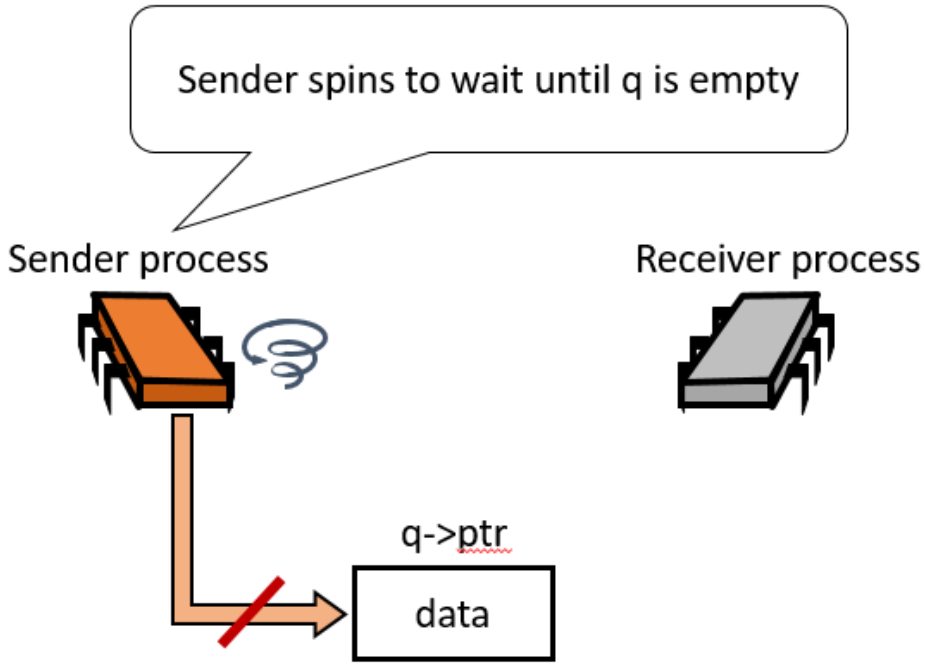
- Race condition

Thread 1	Thread 2	Balance
Read balance: $1000		**$1000**
	Read balance: $1000	**$1000**
	Deposit $200	**$1000**
Deposit $200		**$1000**
Update balance $1000+$200		**$1200**
	Update balance $1000+$200	**$1200**

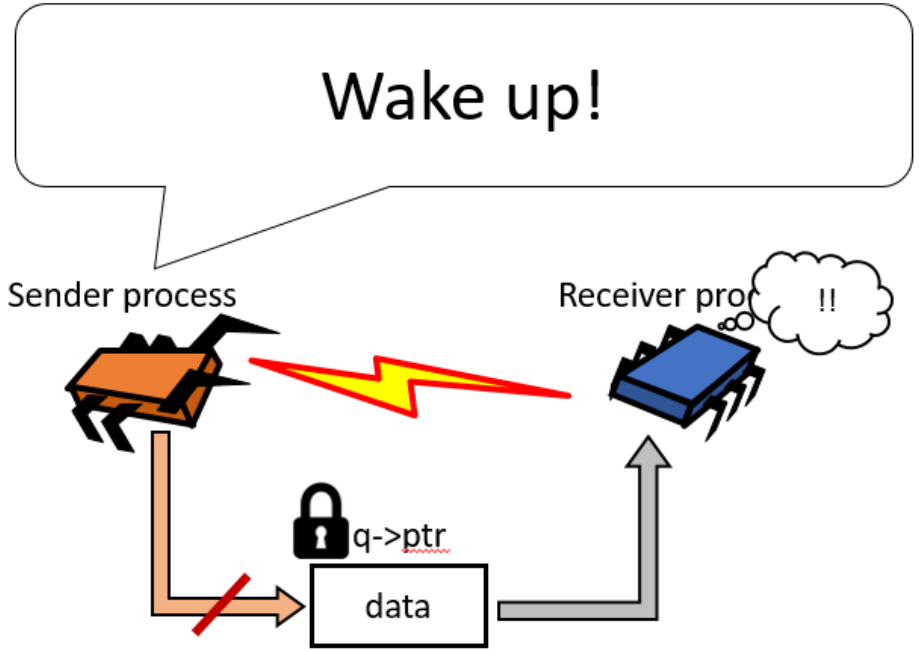
Mutex variables caution

Condition variables

- Spin wait (Spin Lock)

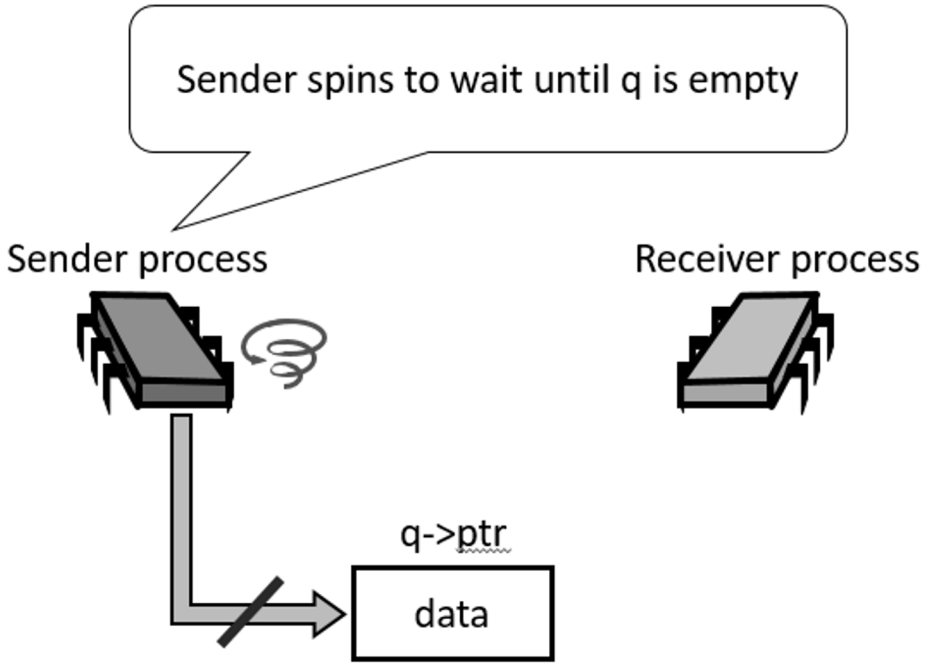


- Sleep – and – WakeUp

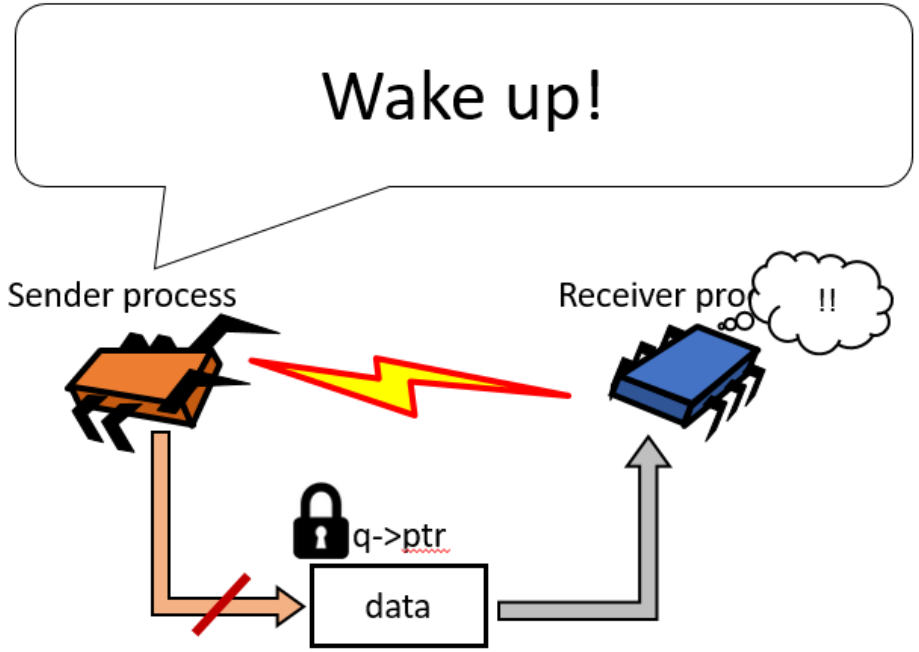


Condition variables

- Spin wait (Spin Lock)



- Sleep – and – WakeUp



Practice

- Normally, performing a merge sort on a single thread takes a long time.
- Implement merge sort with 4 threads executing on 4 different parts. i.e. each thread sorts ¼ of all elements.
- After all child threads are done, the master thread should merge them again to make the whole elements sorted.

- Make sure that multiple processors (at least 4) are assigned to the virtual machine.
- Compare execution time using ‘time’ command.
  - Ex) time ./prac_pthread
- Tip: Use –O3 option to make it significantly faster

Thank you.