Threads and Synchronization Mechanisams

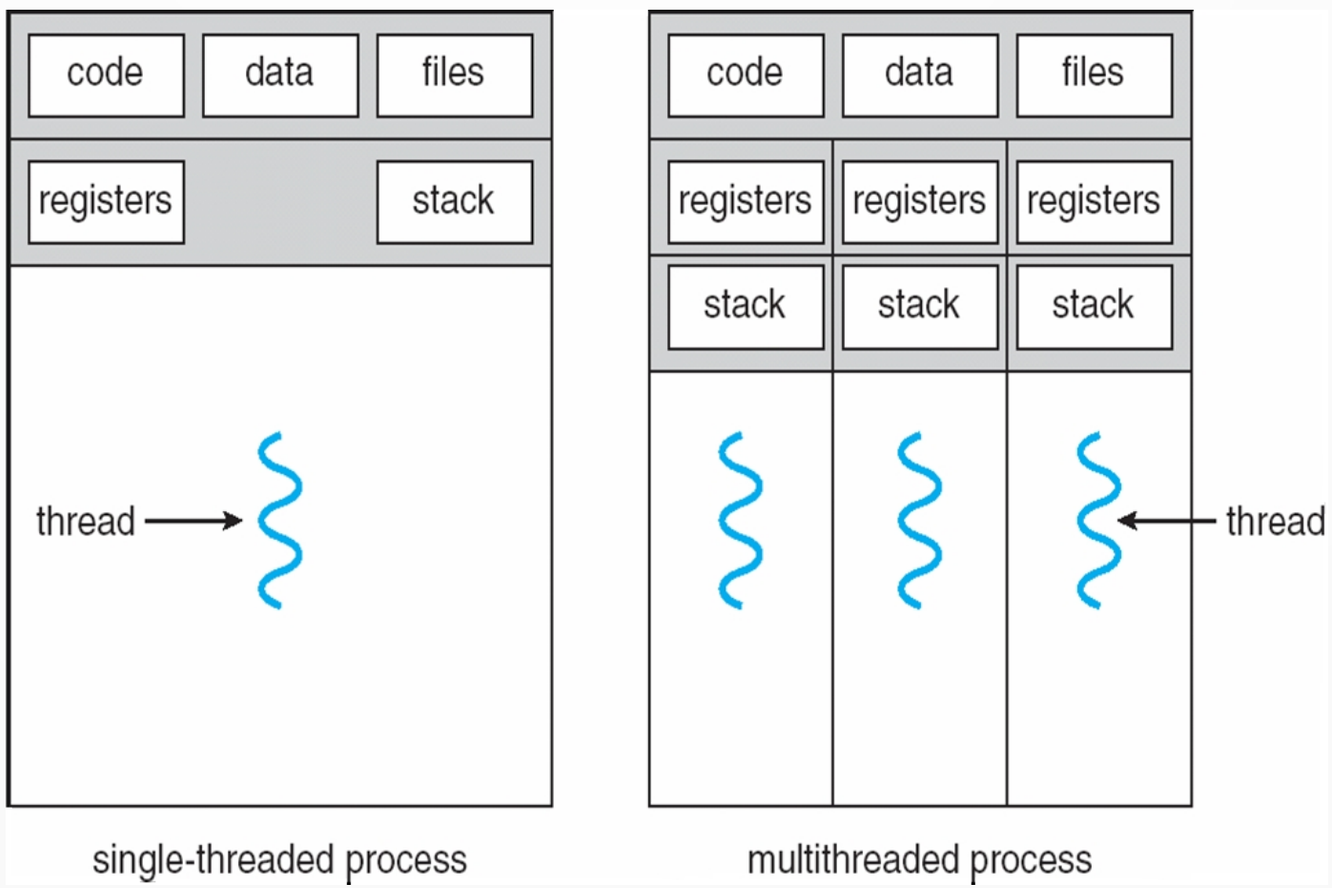
Using POSIX threads

What is a thread?

- Thread is the segment of a process which means a process can have multiple within a process.
- A thread has three states: Running, Ready, and Blocked.
- The thread takes less time to terminate as compared to the process.
- Similar to processes, threads have **TCB**(Thread Control Block) which helps in context switching of threads.

Difference between Single-threaded process and Multi-threaded processes

- Threads uses code ,data,files section of parent process and maintains seperate stack and registers



Why do we need Threads?

- **Responsiveness** - web browsing in one thread, loading images in another
- **Resource sharing** - allow several thread of activity in same address space
- **Economy** - memory & resources for process creation is costly while threads share resources of a process
- **Scalability** - In multiprocessor, threads can run parallel on diff processors, single-thread process can only run on one processor regardless of how many processors exist

How do we programatically implement threads?

- For UNIX like systems, a standardized C language programming interface has been specified by IEEE **POSIX** Threads 1003.1c standard.
- Man man (man page information) for pthredas use man 7 pthreads.
- All the major API's of pthreads can be classified as
  1. Thread Managment(thread creation and destruction)
  - 2.Mutex Locking(Synchronization)
  - 3. Conditional Variables(communication between multiple threads)

Function pointers in c

- What is difference between normal pointers and function pointers ?
- Function pointers can be used in a program to create a function call to functions pointed by them.
- **Syntax** :return type (*ptr_name)(Arg1, Arg2…);
- #include <stdio.h>
- void fun(int a)
- {
  - printf("Value of a is %d\n", a);
- }
- int main()
- { void (*fun_ptr)(int) = &fun;
  - void (*fun_ptr)(int);
  - fun_ptr = &fun;
  - (*fun_ptr)(10);
  - return 0;

  }

Thread Creation and Thread Joining

- All the thread API's declarations psresent in <pthread.h>
- int **pthread_create** (pthread_t* thread_p ,const pthread_attr_t* attr_p ,void* (*start_routine) ,void* arg_p) ;
- Argument 1 ---> stores the thread id.
- Argument2 ---> Thread attributes.
- Argument3 ---> Function to call when thread is invoked.
- Argument4 ---> passing Arguments to a thread.
- pthread_join waits for thread to finish its execution.
- `int pthread_join(pthread_t *thread, void **result);`
- `pthread_exit(EXIT_STATUS) sends the exists status to pthread_join function if it presents.`

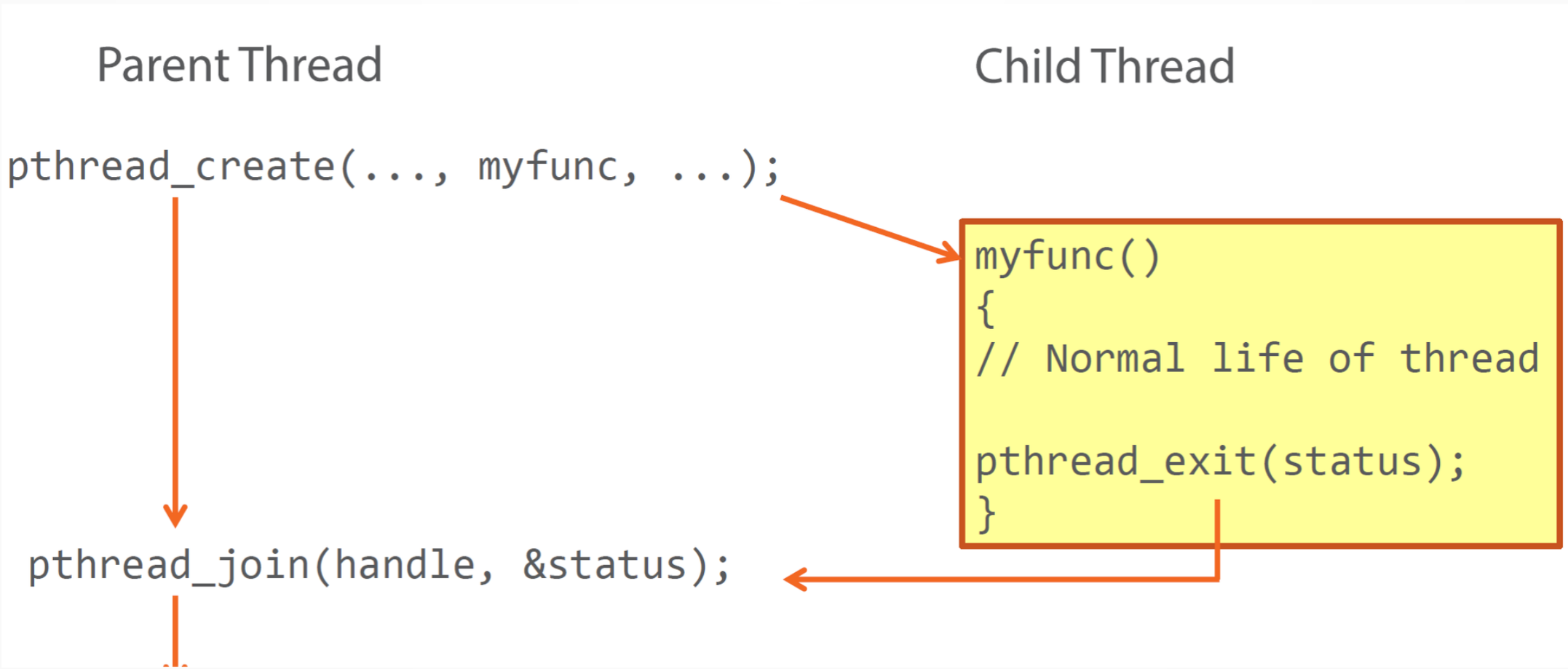
Basic Program to create a Thread

```
#include <stdlib.h>
#include <stdio.h>
#include <pthread.h>
#include <unistd.h>

void* routine() {
    printf("Hello from threads\n");
    sleep(3);
    printf("Ending thread\n");
}

int main(int argc, char* argv[]) {
    pthread_t p1;
    if (pthread_create(&p1, NULL, &routine, NULL) != 0) {
        return 1;
    }

    if (pthread_join(p1, NULL) != 0) {
        return 2;
    }

    return 0;
}
```

How to compile the multi-threaded programs using gcc compiler

- **gcc main.c -o main** will this command works for creating a executabale file?
- **gcc main.c -o main -lpthread** we have use this command for compiling multi threaded programs.Then only linker will able to link the libraries of posix threads successfully.
- What is Makefile concept in linux?
- How to create a make file and how to use this file to compile more than one program at a time?

When will thread completes its execution and what is its life cycle?



Difference between fork() ,exec()and clone() system call

- We can create a new process by calling fork. This system call duplicates the current process (creates a new entry in the process table with same attributes as the current process)
- We can create a new process by calling exec.This system call replaces the current processes.
- clone( ) allows for varying degrees of sharing between the parent and child tasks, controlled by flags like CLONE_FS,CLONE_VM,CLONE_SIGHAND,CLONE_FILES.If no flag is provided it is equivalent to fork.

Running processes parallelly and threads parallelly which more is efficient?

- Using pthread_create() we can create a child thread inside a main thread.
- Similarly using a fork() system we can create a child process inside a parent process.
- This is explained by following two examples **main-processes.c** and **main-threads.c**
- So with multi-threading we can achieve faster performane with usage of minimal resources.

How to run two threads simultaneously

```
#include <stdlib.h>
#include <stdio.h>
#include <pthread.h>
int mails = 0;
void* routine() {
    for (int i = 0; i < 10; i++) {
        mails++;
    }
}

int main(int argc, char* argv[]) {
    pthread_t p1, p2, p3, p4;
    if (pthread_create(&p1, NULL, &routine, NULL) != 0) {
        return 1;
    }
    if (pthread_create(&p2, NULL, &routine, NULL) != 0) {
        return 2;
    }
    if (pthread_join(p1, NULL) != 0) {
        return 3;
    }
    if (pthread_join(p2, NULL) != 0) {
        return 4;
    }
    printf("Number of mails: %d\n", mails);
    return 0;
}
```

How to create thread safe functions in multi threading?

```
#include <stdio.h>
#include <stdlib.h>
#include <unistd.h>
#include <pthread.h>
char * HELLO_MESSAGE;
int THREADS_CREATED = 0;
void * workerThreadFunc(void * tid){
  while(THREADS_CREATED == 0){

  }
  HELLO_MESSAGE = "HELLO WORLD!";
}

void * workerThreadFunc2(void * tid){
    while(THREADS_CREATED == 0){

  }
  for(int i = 0; i < 13; i++){
      printf("\n%c \n",HELLO_MESSAGE[i]);
  }
}

int main(){
  pthread_t tid0;
  pthread_t tid1;
  pthread_create(&tid0,NULL,workerThreadFunc,(void *)&tid0);
  pthread_create(&tid1,NULL,workerThreadFunc2,(void *)&tid1);
  sleep(1);
  THREADS_CREATED = 1;
  pthread_exit(NULL);
  return 0;
}
```

Why do we need to use Synchronization Mechanisams in threads?

- **Issues**:

  1.Data Inconsistency

  2.Loss of Data

  3.Race around condition: Where several process access and manipulate the same data concurrently and the outcome of the execution depends on the particular order. It is called Race condition.

- Yes, we need to use syncronization mechanisams inorder to achieve proper communication between threads.

How to create and run morethan 2 threads simultaneously?

```
#include <stdlib.h>
#include <stdio.h>
#include <pthread.h>

int mails = 0;
pthread_mutex_t mutex;

void* routine() {
    for (int i = 0; i < 10000000; i++) {
        pthread_mutex_lock(&mutex);
        mails++;
        pthread_mutex_unlock(&mutex);
    }
}

int main(int argc, char* argv[]) {
    pthread_t th[8];
    int i;
    pthread_mutex_init(&mutex, NULL);
    for (i = 0; i < 8; i++) {
        if (pthread_create(th + i, NULL, &routine, NULL) != 0) {
            perror("Failed to create thread");
            return 1;
        }
        printf("Thread %d has started\n", i);
    }
    for (i = 0; i < 8; i++) {
        if (pthread_join(th[i], NULL) != 0) {
            return 2;
        }
        printf("Thread %d has finished execution\n", i);
    }
    pthread_mutex_destroy(&mutex);
    printf("Number of mails: %d\n", mails);
    return 0;
}
```

Different types of synchronizing mechanisams used in synchronizing threads

- Majorly we use these four techniques to synchronize thredas.

  1. Mutex Locking
  2. Seamphores
  3. Conditional variables

Mutex Locking Techique

- A Mutex is a Mutually exclusive flag. It acts as a gate keeper to a section of code allowing one thread in and blocking access to all others. This ensures that the code being controlled will only be hit by a single thread at a time.
- Mutex functions and variables

  1. Mutex variable **pthread_mutex_t** var
  2. Mutex locker **pthread_mutex_lock**(&var)
  3. Mutex unlocker **pthread_mutex_unlock**(&var)
  4. Mutex intializer **pthread_mutex_init()**
  5. Mutex destroyer **pthread_mutex_destroy()**

Disadvantages of Multi Threading

- Debugging a multithreaded program is much harder than debugging a single-threaded one, because the interactions between the threads are very hard to control.
- Writing multithreaded programs requires very careful design.
- A program that splits a large calculation into two and runs the two parts as different threads will not necessarily run more quickly on a single processor machine, as there are only one core/thread is running at a time.

References

- https://code-vault.net/course/6q6s9eerd0:1609007479575/lesson/18ec1942c2da46840693efe9b51d86a8
- https://www.intel.in/content/www/in/en/gaming/resources/hyper-threading.html
- https://www.geeksforgeeks.org/thread-in-operating-system/
- https://pages.cs.wisc.edu/~remzi/OSTEP/

THANK YOU