Unix Signals

Name	Default action	Description
SIGHUP	Terminate	Hang up; sent to process when kernel assumes that the user of that process is doing no useful work
SIGINT	Terminate	Interrupt; Control-C
SIGQUIT	Dump	Quit; sent by user to induce halting of process and production of core dump
SIGILL	Dump	Illegal instruction
SIGTRAP	Dump	Trace trap; triggers the execution of code for process tracing
SIGIOT	Dump	IOT instruction
SIGEMT	-	EMT instruction
SIGFPE	Dump	Floating-point exception
SIGKILL	Term	Kill; terminate process
SIGBUS	Dump	Bus error
SIGSEGV	Dump	Segmentation violation: process attempts to access location outside its virtual address space
SIGSYS	Dump	Bad argument to system call
SIGPIPE	Terminate	Write on a pipe that has no readers attached to it
SIGALRM	Terminate	Alarm clock: issued when a process wishes to receive a signal after a period of time
SIGTERM	Terminate	Software termination
SIGUSR1	Terminate	User-defined signal 1
SIGUSR2	Terminate	User-defined signal 2
SIGCHLD	Ignore	Death of a child
SIGCONT	Continue	Resume execution, if stopped
SIGSTOP	Stop	Stop process execution, Ctrl-Z
SIGPWR	Terminate	Power failure

Mutex in pthread

How to create?

```
pthread_mutex_t mutex;
```

How to lock and unlock?

```
pthread_mutex_lock(&mutex); // make sure to pass the reference to the mutex.
```

critical section

```
pthread_mutex_unlock(&mutex);
```

At the end of the program destroy the mutex by calling

```
pthread_mutex_destroy(&mutex);
```

Semaphores in pthread.

You need to include: `#include <semaphore.h>`

How to create and initialize?

```
sem_t semaphore;
sem_init(&semaphore, 0, initial_value);
```

//first argument: reference to the semaphore,

//second argument: 0 indicates that semaphore is shared between threads of the process therefore semaphore should be located at some address that is visible to all the threads in the process.

If this value is nonzero, then it means semaphore is shared between processes, and it should be located somewhere that is visible to all the processes.

// Third argument: This is the initial starting value

How to use sem_wait and sem_signal and destroy the semaphore? Use the sem_wait() and sem_post() methods.

```
sem_wait(&semaphore);
sem_post(&semaphore);
sem_destroy(&semaphore);
```

How to fork a process?

```
int pid = fork();
```

This method returns the pid of the child to the parent and 0 to the child process.

How to create a pthread?

1. **First create a function for the thread to run. Make sure that it returns a void pointer.**
    a. **This can only take one argument.**
2. **Create a pthread_t variable for the thread.**
    a. **pthread_t thread1;**
3. **Then use the pthread_create(pthread_t*, pthread_attr_t*, void*(* start_routine)(void *))**
    a. **Basically, we need to pass a reference to the pthread_t variable, NULL for pthread attribute, function for the thread to run, and the only argument for the pthread function.**
    b. **Ex: pthread_create(&threads[i], NULL, increment_counter, (void *) arg)**
    c. **Ex: pthread_create(&threads[i], NULL, some_func, NULL) // this function does not take any argument.**
4. **Make sure that you call the pthread_join(pthread_t, void** retVlaue);**
    a. **This method makes sure that the thread that creates the new thread waits until the new thread finishes, if you do not do this, and the main thread finishes processing, new thread that you created automatically dies.**
    b. **The second argument is for keeping track of the return value of the thread.**