Day1

Table of Contents

1. Agenda

- What is thread
- What are Pthreads
- Pthreads Overview
- Why Pthreads
- Designing Threaded Programs
- Pthreads APIs
- Creating and Terminating Threads

2. Terminologies to remember

- threads
- pthread
- process
- context
- context switching
- concurrency
- parallelism
- multithreading
- cores
- hypterthreading

3. Pthread Hello World

3.1. code

```c
#include<stdio.h>
#include<pthread.h>
void* hello(){
    printf("Hello, World\n");
}
int main(){
    pthread_t t;
    pthread_create(&t, NULL, hello, NULL);
    pthread_join(t, NULL);
    return 0;
}
```

3.2. compile

```
gcc pth1.c -o pth1.out -lpthread
```

3.3. run

```
./pth1.out
```

```
Hello, World
```

4. creating 2 threads

4.1. code

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

void* task1(){
    printf("starting task1\n");
    sleep(10);
    printf("ending task1\n");
}

void* task2(){
    printf("starting task2\n");
    sleep(10);
    printf("ending task2\n");
}
int main(){
    pthread_t t1, t2;
    pthread_create(&t1, NULL, task1, NULL);
    pthread_create(&t2, NULL, task2, NULL);
    pthread_join(t1, NULL);
    pthread_join(t2, NULL);
    return 0;
}
```

4.2. compile

```
gcc twoThreads.c -lpthread
```

4.3. run

```
./a.out
```

```
starting task1
starting task2
ending task1
ending task2
```

5. creating N number of threads

5.1. code

```c
#include<stdio.h>
#include<unistd.h>
#include<pthread.h>
#define N 10

void* task(){
    printf("starting task\n");
    sleep(5);
    printf("ending task\n");
}
int main(){
    pthread_t t[N];
    for(int i = 0; i < N; i++){
        pthread_create(&t[i], NULL, task, NULL);
    }

    for(int i = 0; i < N; i++){
        pthread_join(t[i], NULL);
    }
    return 0;
}
```

5.2. compile

```
gcc nthreads.c -o nthreads.out -lpthread
```

5.3. run

```
./nthreads.out
```

```
starting task
starting task
starting task
starting task
starting task
starting task
starting task
starting task
starting task
starting task
ending task
ending task
ending task
ending task
ending task
ending task
ending task
ending task
ending task
ending task
```

6. Devide two tasks between equal number of threads

6.1. code

```c
#include<stdio.h>
#include<unistd.h>
#include<pthread.h>
#define N 20

void* task1(){
```

```
    printf("starting task1\n");
    sleep(2);
    printf("ending task1\n");
}

void* task2(){
    printf("starting task2\n");
    sleep(2);
    printf("ending task2\n");

}

int main(){
    pthread_t t1[N];
    for(int i = 0; i < N; i++){
        if(i < N/2)
            pthread_create(&t1[i], NULL, task1, NULL);
        else
            pthread_create(&t1[i], NULL, task2, NULL);


    }

    for(int i = 0; i < N; i++){
        pthread_join(t1[i], NULL);
    }
    return 0;
}
```

6.2. compile

```
gcc devideTaskBetweenThreads.c -lpthread
```

6.3. run

```
./a.out
```

```
starting task1
starting task1
```

```
starting task1
starting task1
starting task1
starting task1
starting task1
starting task1
starting task1
starting task1
starting task1
starting task2
starting task2
starting task2
starting task2
starting task2
starting task2
starting task2
starting task2
starting task2
starting task2
ending task1
ending task1
ending task1
ending task1
ending task1
ending task2
ending task2
ending task1
ending task1
ending task1
ending task1
ending task1
ending task2
ending task2
ending task2
ending task2
ending task2
ending task2
ending task2
ending task2
ending task2
```

Author: Abhishek Raj
Created: 2024-12-11 Wed 23:05