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. Scripts

2.1. compile script

```sh
#!/bin/sh

inputFile=$1
outputFile="${1%.*}.out"      # extract the name of the file without extension and adding
#cmd=`mpicc $inputFile -o $outputFile`
cmd="gcc $inputFile -o $outputFile -lpthread"     # running code using MPI
echo "------------------------------------------------------------------"
echo "Command executed: $cmd"
echo "------------------------------------------------------------------"
$cmd
echo "Compilation successful. Check at $outputFile"
echo "------------------------------------------------------------------"
```

2.2. run script

```sh
#!/bin/sh

help(){
    echo
    echo "Usage: submit.sh -[flags]"
    echo "-h                : help"
    echo "-f <filename>     : enter file"
    echo "-t <threads>      : number of threads"
    echo "-c <compile>      : first compile"
    echo "-n <iterations>   : number of times to execute"
    echo
    echo "eg: bash run.sh -f main.c -t 8 -c -n 3"
    echo
}

isCompiled=0
nthreads=8

while getopts 'n:f:t:hc' flag; do
  case "${flag}" in
    t) nthreads="${OPTARG}" ;;
    f) file=${OPTARG} ;;
    c) isCompiled=1 ;;
    n) iterations="${OPTARG}" ;;
    h) help
        exit 1 ;;
    *) echo "Invalid flag...(-h for help)"
       exit 1 ;;
  esac


done

if [[ -z $file ]]; then
    echo "You forgot to mention filename!!!"
    exit 1
fi
if [[ -z $iterations ]]; then
    iterations=1
fi



outputFile="${file%.*}.out"      # extract the name of the file without extension and add

if [[ $isCompiled == 1 ]]; then
    gcc $file -o $outputFile -lpthread -lm
fi

export OMP_NUM_THREADS=$nthreads
```

```
echo "--------------------------------------------------------"
i=1
while [ 1 == 1 ];
do
    ./$outputFile
    if [[ $i == $iterations ]]; then
        exit 1
    fi
    i=$(( $i + 1 ))
    echo "--------------------------------------------------------"
done
```

3. Pthread Hello World

3.1. code

```
#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 or more threads

4.1. code

```
#include<stdio.h>
#include<stdlib.h>
#include<pthread.h>
#define N 2
void *hello(void* threadId){
    long tid = (long)threadId;
    printf("Hello World %ld of %d\n", tid, N);
    return NULL;
}
int main(){
    pthread_t t1, t2;

    pthread_create(&t1, NULL, hello, (void*) 0);
    pthread_join(t1, NULL);
    pthread_create(&t2, NULL, hello, (void*) 1);
    pthread_join(t2, NULL);
    return 0;
}
```

4.2. compile

```
file=pth2
gcc $file.c -o $file.out -lpthread
#bash compile.sh $file.c
```

4.3. run

```
file=pth2
./$file.out
```

```
Hello World 0 of 2
Hello World 1 of 2
```

5. Creating N number of threads

5.1. code

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

void *hello(void* threadId){
    long tid = (long)threadId;

    printf("Hello World %ld of %d\n", tid, N);
    return NULL;
}

int main(){
    pthread_t* t;
    t = malloc(sizeof(pthread_t) * N);

    for(long i = 0; i < N; i++)
        pthread_create(&t[i], NULL, hello, (void*)i);
    for(long i = 0; i < N; i++)
        pthread_join(t[i], NULL);
    free(t);
    return 0;
}
```

5.2. compile

```
file=pth3
gcc $file.c -o $file.out -lpthread
```

5.3. run

```
file=pth3
./$file.out
```

```
Hello World 0 of 10
Hello World 1 of 10
Hello World 2 of 10
Hello World 3 of 10
Hello World 4 of 10
Hello World 5 of 10
Hello World 6 of 10
Hello World 7 of 10
Hello World 8 of 10
Hello World 9 of 10
```

6. Task1

Initialize an array of size N and assign each thread to print element which is equal to their threadId.

```
Thread 0 is printing 0
Thread 1 is printing 1
thread 2 is printing 2
.
.
.
Thread N is printing N
```

The output can be in any order.

7. Solution task1

```
#include<stdio.h>
#include<stdlib.h>
#include<unistd.h>
#include<pthread.h>
#define N 16
int arr[N];
void* hello(void* threadId){
    long tid = (long)threadId;
    printf("Thread id %ld is printing %d\n", tid, arr[tid]);
    return NULL;
}
int main(){
    pthread_t t[N];
    for(int i = 0; i < N; i++){
        arr[i] = i;
    }
    for(long i = 0; i < N; i++){
        pthread_create(&t[i], NULL, hello, (void*) i);
    }
    for(long i = 0; i < N; i++){
        pthread_join(t[i], NULL);
    }
    return 0;
}
```

```
Thread id 2 is printing 2
Thread id 0 is printing 0
Thread id 8 is printing 8
Thread id 1 is printing 1
Thread id 9 is printing 9
Thread id 3 is printing 3
Thread id 6 is printing 6
Thread id 10 is printing 10
Thread id 4 is printing 4
Thread id 11 is printing 11
Thread id 12 is printing 12
Thread id 5 is printing 5
Thread id 7 is printing 7
Thread id 13 is printing 13
Thread id 14 is printing 14
Thread id 15 is printing 15
```

8. test1

```c
#include<stdio.h>
#include<unistd.h>
#include<pthread.h>
#define N 2
void* hello(void* threadId){
    long tid = (long)threadId;
    printf("Forking %ld of %d\n", tid, N);
    sleep(1);
    printf("Joining %ld of %d\n", tid, N);
    return NULL;
}
int main(){
    pthread_t t1, t2, t3, t4;
    pthread_create(&t1, NULL, hello, (void*) 0);
    pthread_create(&t2, NULL, hello, (void*) 1);
    pthread_create(&t3, NULL, hello, (void*) 2);
    pthread_create(&t4, NULL, hello, (void*) 3);
    pthread_join(t1, NULL);
    pthread_join(t2, NULL);
    pthread_join(t3, NULL);
    pthread_join(t4, NULL);
    return 0;
}
```

```
Forking 0 of 2
Forking 1 of 2
Forking 2 of 2
Forking 3 of 2
Joining 2 of 2
Joining 1 of 2
Joining 0 of 2
Joining 3 of 2
```

9. test2

```c
#include<stdio.h>
#include<stdlib.h>
#include<unistd.h>
#include<pthread.h>
#define N 16
void* hello(void* threadId){
    long tid = (long)threadId;
    printf("Forking %ld of %d\n", tid, N);
    sleep(1);
    printf("Joining %ld of %d\n", tid, N);
    return NULL;
}
int main(){
    pthread_t t[N];

    for(long i = 0; i < N; i++){
        pthread_create(&t[i], NULL, hello, (void*) i);
    }
    for(long i = 0; i < N; i++){
        pthread_join(t[i], NULL);
    }
    return 0;
}
```

```
Forking 0 of 16
Forking 3 of 16
Forking 1 of 16
Forking 2 of 16
Forking 4 of 16
Forking 5 of 16
Forking 6 of 16
Forking 7 of 16
Forking 8 of 16
Forking 9 of 16
Forking 10 of 16
Forking 11 of 16
Forking 12 of 16
Forking 13 of 16
Forking 14 of 16
Forking 15 of 16
Joining 2 of 16
Joining 0 of 16
Joining 3 of 16
Joining 5 of 16
Joining 1 of 16
Joining 6 of 16
Joining 7 of 16
Joining 4 of 16
Joining 9 of 16
Joining 8 of 16
Joining 10 of 16
Joining 11 of 16
Joining 12 of 16
Joining 13 of 16
Joining 14 of 16
Joining 15 of 16
```

Author: Abhishek Raj
Created: 2024-06-25 Tue 15:31