Thread Programming

Faculty of Computer Science and Engineering
Ho Chi Minh City University of Technology
Introduction
Introduction

<table>
<thead>
<tr>
<th>CODE</th>
<th>DATA</th>
<th>STACK</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td></td>
<td></td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>CODE</th>
<th>DATA</th>
<th>STACK</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td></td>
<td></td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>CODE</th>
<th>DATA</th>
<th>STACK</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td></td>
<td></td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>CODE</th>
<th>DATA</th>
<th>STACK</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td></td>
<td></td>
</tr>
</tbody>
</table>
Introduction

- POSIX (Portable Operating System Interface) thread
  - popular in *NIX system
  - API and user-level thread libraries
- Sun Thread
Create thread

```c
#include <pthread.h>
int pthread_create(
    pthread_t *thread,
    pthread_attr_t *attr,
    void **(*start_routine)(void *),
    void *arg);
```

- Return value:
  - 0 : if successful
  - <>0: otherwise
Create thread - arguments

- Third argument `start_routine`
  - pointer to a function
  - return type is `void*`
  - argument type is `void*`, this argument is passed as the fourth argument of this function

- Fourth argument `arg`
  - passed to function `start_routine`
Exit thread

- Thread exits when:
  - `start_routine` exits
  - `pthread_exit()` is called
  - thread is interrupted by `pthread_cancel()`
  - process terminates
  - one of threads calls system call `exec()`

- Exit thread
  ```c
  void pthread_exit(void* retval)
  ```
```c
#include <pthread.h>
#include <stdio.h>
void* func(void* arg)
{
    int i;
    for (i = 0; i < 2; i++) {
        printf("This is thread %d\n", *((int*)arg));
        sleep (1);
    }
}
int main (int argc, char **argv) {
    int i;
    pthread_t tid[3];
    for (i=0; i<3; i++){
        pthread_create(&tid[i], NULL, func, (void*)&tid[i]);
    }
    sleep (5);
    return 0;
}
```
Compile and execute

$ gcc pthcreate.c -o pthcreate -lpthread
$ ./pthcreate
This is thread -1208886368
This is thread -1219376224
This is thread -1229866080
This is thread -1208886368
This is thread -1219376224
This is thread -1229866080
Wait for a thread

```c
#include <pthread.h>
int pthread_join(pthread_t th,
        void **thread_return);
```

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

void* func(void* arg) {
    int i;
    for (i = 0; i < 2; i++) {
        printf("This is thread %d\n",*(((int*)arg)));
        sleep(1);
    }
}

int main(int argc, char **argv) {
    int i; pthread_t tid[3];
    for (i=0; i<3; i++) {
        pthread_create(&tid[i],NULL, func, (void*)&tid[i]);
        pthread_join(tid[i], NULL);
    }
    return 0;
}
```
Compile and execute

$gcc pthjoin.c -o pthjoin -lpthread
$./pthjoin
This is thread -1208710240
This is thread -1208710240
This is thread -1208710240
This is thread -1208710240
This is thread -1208710240
This is thread -1208710240
This is thread -1208710240
Pass data to thread

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

struct char_printParms {
    char character;
    int count;
};

void* char_print (void* args) {
    struct char_printParms* p = 
        (struct char_printParms*) args;
    int i;
    for (i=0; i<p->count; i++)
        printf("%c\n", p->character);
    return NULL;
}
```
Pass data to thread (cont)

```c
int main () {
    pthread_t tid;
    struct char_printParms th_args;
    th_args.character = 'X';
    th_args.count = 5;
    pthread_create(&tid, NULL, &char_print, &th_args);
    pthread_join (tid, NULL);
    return 0;
}
```
Compile and execute

$ gcc charprint.c -o charprint -lpthread
$ ./pthjoin

X

X cuu duong than cong . com

X

X cuu duong than cong . com
Questions???