POSIX-thread Intro

Outline

- Parallel Programming Models
- Pthreads introduction
- Introduction of SystemCall
- Introduction of Scheduler setting
- Experimental Environment Introduction
- Project 1

Parallel Programming Models

- An abstraction of parallel computer architecture.
- Classification
  - Process interaction
    - Shared memory
    - Message passing
  - Problem decomposition
    - Task
    - Data

Parallel Programming Models

Class of interaction	Class of decomposition	Example implementations
Shared memory	Task	Apache Giraph
Shared memory	Data	Openmp, CUDA, **Pthread**
Message passing	Task	Verilog, VHDL

Pthreads Introduction

- The Pthreads provides API for developing parallel program on shared memory architecture.
- The Pthreads is for C/C++.

Pthreads Introduction

- #include <pthread.h>
- pthread_create()
- pthread_join()

Pthreads Introduction

- pthread_create() : start a new thread

```
int pthread_create(pthread_t *thread, const pthread_attr_t *attr,
                   void *(*start_routine) (void *), void *arg);
```

- *thread : point to the thread object you create
- *attr : define NULL
- *start_routine : point to the function that we want to run
- *arg : a pointer that will be given to our function

Pthreads Introduction

- pthread_join() : Allows the calling thread to wait for the ending of the target thread.

```
int pthread_join( pthread_t thread,
                  void **   retval
);
```

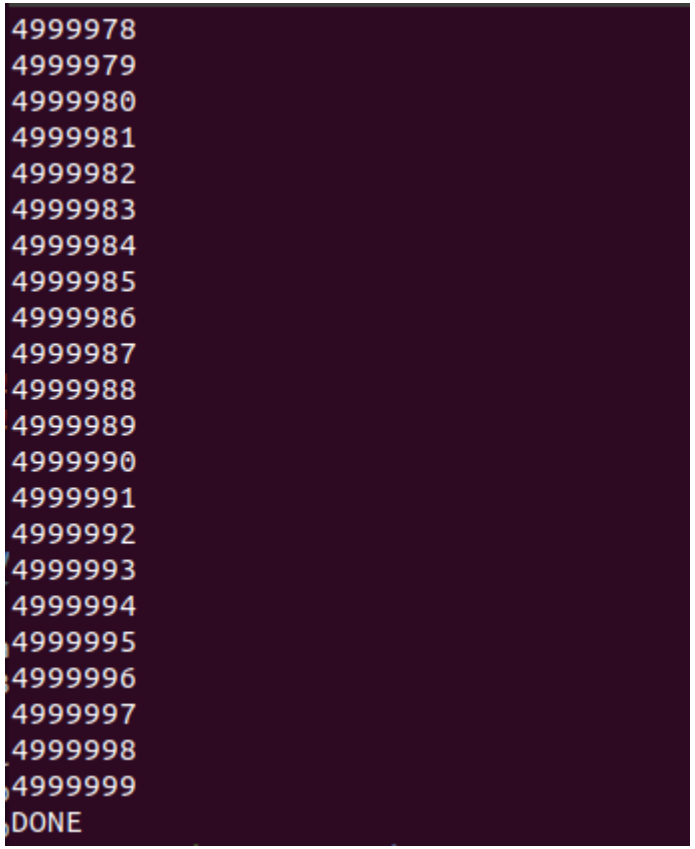
- thread : point to the thread object you create
- retval : return value, else NULL

Using Pthreads(top)

```
top - 23:31:21 up 6 days, 11:16,  1 user,  load average: 1.20, 0.80, 0.37
Threads: 676 total,   5 running, 592 sleeping,   0 stopped,   0 zombie
%Cpu(s):  5.6 us, 39.0 sy,  0.0 ni, 54.9 id,  0.0 wa,  0.0 hi,  0.5 si,  0.0 st
KiB Mem :  8141488 total,  4849280 free,  1815552 used,  1476656 buff/cache
KiB Swap:   969960 total,   969960 free,        0 used.  6002280 avail Mem

  PID USER      PR  NI    VIRT    RES    SHR S %CPU %MEM     TIME+ COMMAND
20370             20   0  844948  57724  31296 R 50.5  0.7   1:10.70 gnome-term+
22469             20   0   88616    824    744 R 49.2  0.0   0:02.26 Prac1.out
22468             20   0   88616    824    744 R 43.2  0.0   0:02.09 Prac1.out
22331             20   0       0      0      0 I 28.9  0.0   0:09.44 kworker/u2+
21688             20   0       0      0      0 I 15.0  0.0   0:08.67 kworker/u2+
22465             20   0       0      0      0 I 14.3  0.0   0:01.36 kworker/u2+
 1559             20   0 3676276 229904  86520 S  3.3  2.8  29:12.71 gnome-shell
 1416             20   0  534868 100660  51980 S  2.3  1.2  17:31.46 Xorg
22461             20   0   51860   4512   3276 R  0.7  0.1   0:00.65 top
 1654             20   0  319584   9744   8636 S  0.3  0.1   0:00.32 gmain
 1767             20   0  549052  48008  35164 S  0.3  0.6  10:18.91 vmtoolsd
21843             20   0       0      0      0 I  0.3  0.0   0:10.04 kworker/u2+
    1             20   0  225436   9212   6700 S  0.0  0.1   0:18.74 systemd
    2             20   0       0      0      0 S  0.0  0.0   0:00.11 kthreadd
    3              0 -20       0      0      0 I  0.0  0.0   0:00.00 rcu_gp
    4              0 -20       0      0      0 I  0.0  0.0   0:00.00 rcu_par_gp
    6              0 -20       0      0      0 I  0.0  0.0   0:00.00 kworker/0:+
```

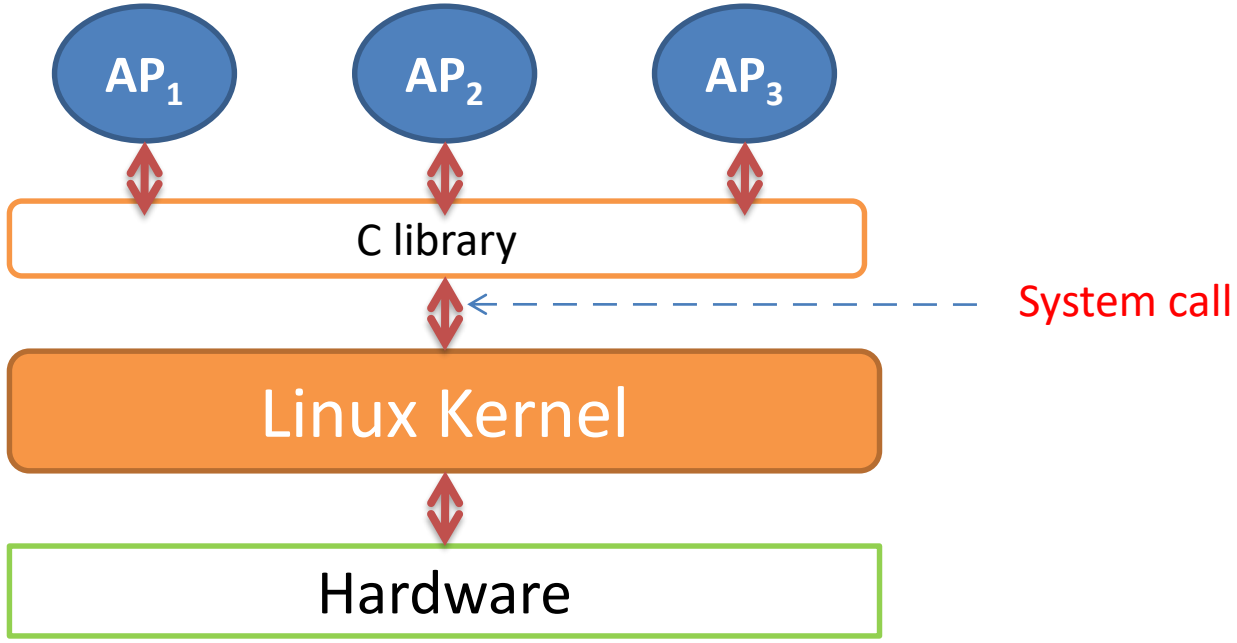
Using Pthreads(with join)



System Call

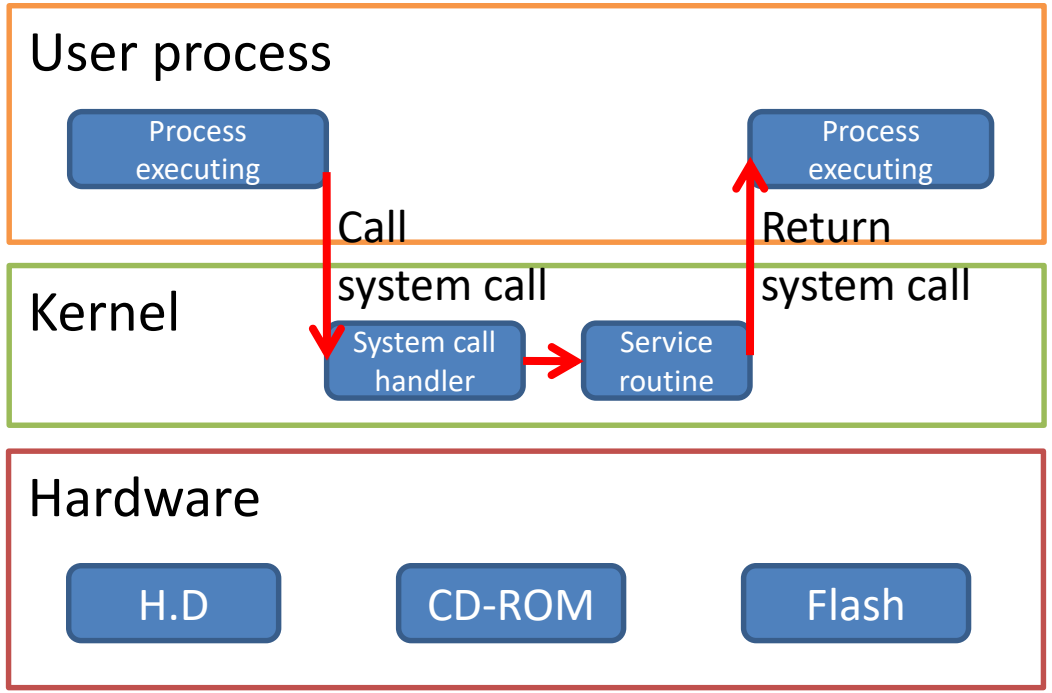
Definition:

System calls provide an interface between the hardware and user-space processes.



System Call

- Example:



Scheduler setting

Mostly used function:

- We can use functions below to set the state of cores
  - CPU_ZERO(&set)
  - CPU_SET(i, &set)

Example:

```
#define _GNU_SOURCE

cpu_set_t  set;
CPU_ZERO(&set);              //disable all CPUs
CPU_SET(0, &set);            //enable CPU#0
CPU_SET(2, &set);            //enable CPU#2
```

Scheduler setting

- sched_setaffinity(*pid_t pid, size_t cpusetsize, const cpu_set_t *mask*)
  - We can use this function to set the CPU affinity mask of the thread whose ID is pid to the value specified by mask
- cpu_alloc
  - Allocate a CPU set.
- sched_getcpu()
  - We can use it to get CPU's ID that the specific thread runs on.

Experimental Environment

- PC : at least 4 cores
- OS : Ubuntu 16.04或以上
- Compiler g++ 5.4.0

Ubuntu 16.04

- Virtual Machine recommended
  - Oracle
  - VMware
- Download & Install information can be found on Google