CSGE602055 Operating Systems
CSF2600505 Sistem Operasi
Minggu 06: Concurency: Processes & Threads

Rahmat M. Samik-Ibrahim

Universitas Indonesia

http://rms46.vlsm.org/2/207.html

REV088 25-Oct-2017

OS172 | INT TU/TH 13:00-15:00 | EXT TH 19:00-21:50

Minggu 00	29 Aug - 05 Sep 2017	Intro & Review
Minggu 01	07 Sep - 12 Sep 2017	IPR, SED, AWK, REGEX, & Scripting
Minggu 02	14 Sep - 19 Sep 2017	Protection, Security, Privacy, & C-language
Minggu 03	26 Sep - 30 Sep 2017	BIOS, Loader, Systemd, & I/O
Minggu 04	03 Okt - 07 Okt 2017	Addressing, Shared Lib, Pointer & I/O Programming
Minggu 05	10 Okt - 14 Okt 2017	Virtual Memory
Ming. UTS	15 Okt - 24 Okt 2017	
Minggu 06	26 Okt - 31 Okt 2017	Concurency: Processes & Threads
Minggu 07	02 Nov - 07 Nov 2017	Synchronization
Minggu 08	09 Nov - 14 Nov 2017	Scheduling & Network Sockets Programming
Minggu 09	16 Nov - 21 Nov 2017	File System & Persistent Storage
Minggu 10	23 Nov - 28 Nov 2017	Special Topic: Blockchain
Cadangan	30 Nov - 09 Des 2017	
Ming. UAS	10 Des - 23 Des 2017	

Agenda

Week 06: Processes & Threads

- Reference: (OSCE2e ch3/4) (UCB 02 03) (UDA P2L1/2/3) (OLD 03)
- Process Concept
  - Program (passive) $\leftrightarrow$ Process (active)
  - Process in Memory: $|$ *Stack* $\cdots$ *Head* $|$ *Data* $|$ *Text* $|$
  - Process State: $|$ *running* $|$ *waiting* $|$ *ready* $|$
  - `fork()` and `execlp()`
- The Multi-process Synchronization Problem
  - Producer-Consumer (Bounded Buffer)
  - Readers-Writers
  - Dining Philosopher
- Communication
  - Pipes
  - Sockets
  - RPC

Thread

- Multicore Programming
- Multithreading Models
- Threading Issues
- Benefits
    - Responsiveness
    - Resource Sharing
    - Economy
    - Scalability
- Concurrency vs. Parallelism
- Multithreading Models
    - Many to One
    - One to One
    - Many to Many
    - Multilevel Models
- Pthreads

Makefile

```
CC=gcc
P00=00-fork
P01=01-fork
...
P16=16-fork
P17=17-exec

EXECS= \
   $(P00) \
   $(P01) \
   ....
   $(P16) \
   $(P17) \

all:  $(EXECS)

$(P00): $(P00).c
   $(CC) $(P00).c -o $(P00)

$(P01): $(P01).c
   $(CC) $(P01).c -o $(P01)

...

$(P16): $(P16).c
   $(CC) $(P16).c -o $(P16)

$(P17): $(P17).c
   $(CC) $(P17).c -o $(P17)

clean:
   rm -f $(EXECS)
```

00-fork

```
/*
 * (c) 2016-2017 Rahmat M. Samik-Ibrahim
 * http://rahmatm.samik-ibrahim.vlsm.org/
 * This is free software.
 * REV01 Wed Oct 25 20:13:15 WIB 2017
 * START Mon Oct 24 09:42:05 WIB 2016
 */

#include <stdio.h>
#include <unistd.h>
#include <sys/types.h>

void main(void) {
   printf("Start:  PID[%d] PPID[%d]\n", getpid(), getppid());
}

>>>>> $ 00-fork
Start:  PID[2279] PPID[1448]
```

01-fork

```
/*
 * (c) 2016-2017 Rahmat M. Samik-Ibrahim
 * http://rahmatm.samik-ibrahim.vlsm.org/
 * This is free software.
 * REV02 Wed Oct 25 20:14:24 WIB 2017
 * START Mon Oct 24 09:42:05 WIB 2016
 */
#include <stdio.h>
#include <unistd.h>
#include <sys/types.h>
char string[256];

void main(void) {
   char *iAM="PARENT";

   printf("PID[%d] PPID[%d] (START:%s)\n", getpid(), getppid(), iAM);
   if (fork() > 0) {
      sleep(1);
      printf("PID[%d] PPID[%d] (IFF0:%s)\n", getpid(), getppid(), iAM);
   } else {
      iAM="CHILD";
      printf("PID[%d] PPID[%d] (ELSE:%s)\n", getpid(), getppid(), iAM);
   }
   printf("PID[%d] PPID[%d] (STOP:%s)\n", getpid(), getppid(), iAM);
}
```

```
>>>>> $ 01-fork
PID[2285] PPID[1448] (START:PARENT)
PID[2286] PPID[2285] (ELSE:CHILD)
PID[2286] PPID[2285] (STOP:CHILD)
PID[2285] PPID[1448] (IFF0:PARENT)
PID[2285] PPID[1448] (STOP:PARENT)
```

02-fork

```
>>>>> $ diff 01-fork.c 02-fork.c
21d20
<         sleep(1);
25a25
>         sleep(1);
>>>>> $ 01-fork
PID[2528] PPID[1448] (START:PARENT)
PID[2529] PPID[2528] (ELSE:CHILD)
PID[2529] PPID[2528] (STOP:CHILD)
PID[2528] PPID[1448] (IFF0:PARENT)
PID[2528] PPID[1448] (STOP:PARENT)
>>>>> $ 02-fork
PID[2530] PPID[1448] (START:PARENT)
PID[2530] PPID[1448] (IFF0:PARENT)
PID[2530] PPID[1448] (STOP:PARENT)
PID[2531] PPID[2530] (ELSE:CHILD)
>>>>> $ PID[2531] PPID[1] (STOP:CHILD)

>>>>> $
```

03-fork

```
>>>>> $ diff 01-fork.c 03-fork.c
10d9
<
13a13
> #include <sys/wait.h>
21c21
<         sleep(1);
---
>         wait(NULL);
>>>>> $ 01-fork
PID[2596] PPID[1448] (START:PARENT)
PID[2597] PPID[2596] (ELSE:CHILD)
PID[2597] PPID[2596] (STOP:CHILD)
PID[2596] PPID[1448] (IFF0:PARENT)
PID[2596] PPID[1448] (STOP:PARENT)
>>>>> $ 03-fork
PID[2598] PPID[1448] (START:PARENT)
PID[2599] PPID[2598] (ELSE:CHILD)
PID[2599] PPID[2598] (STOP:CHILD)
PID[2598] PPID[1448] (IFF0:PARENT)
PID[2598] PPID[1448] (STOP:PARENT)
```

X

```
X
```

The End

- This is the end of the presentation.