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

Rahmat M. Samik-Ibrahim

Universitas Indonesia

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

REV089 26-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

1. Start
2. Agenda
3. Week 06
4. Process Map
5. Process State
6. Makefile
7. 00-fork
8. 01-fork
9. 02-fork
10. 03-fork
11. 01-fork vs 02-fork vs 03-fork
12. 04-sleeping
13. 05-fork
14. X
15. The End

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* |
  - Process Control Block (PCB)
  - (I/O vs CPU) Bound Processes
- Process Creation
  - PID: Process Identifier (uniq)
  - The Parent Process forms a tree of Children Processes
  - `fork()`, new process system call (clone)
  - `execlp()`, replaces the clone with a new program.
- Process Termination
  - `wait()`, until the child process is terminated.

Process Map

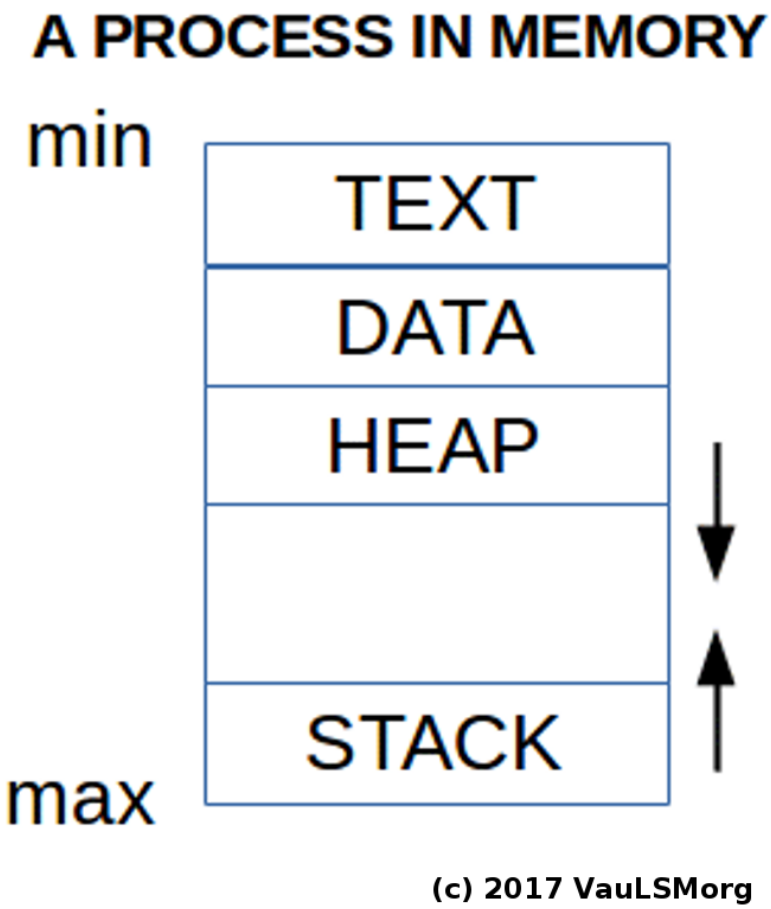

Figure: A Process in Memory

Process State

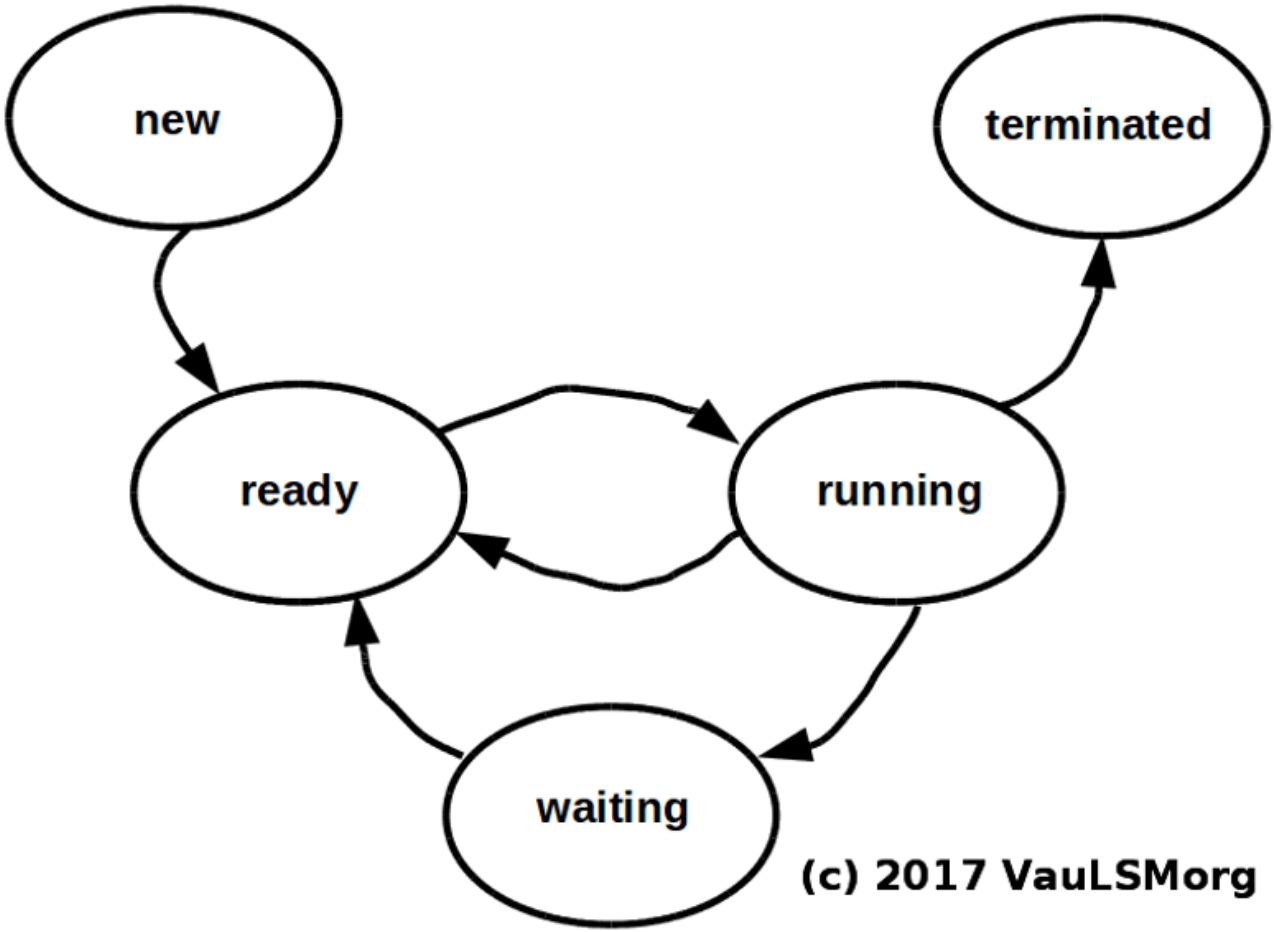

Figure: A Process State

More about Process

- Android Systems
  - Dalvik VM Performance Problem: Replaced with ART (Android Runtime).
  - Foreground Processes: with an User Interface (UI) for Videos, Images, Sounds, Texts, etc.
  - Background Processes: with a service with no UI and small memory footprint.
- 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
  - Parallelism on a multi-core system.
- 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("00-fork:  PID[%d] PPID[%d]\n", getpid(), getppid());
}

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

01-fork

```
>>>>> $ cat 01-fork.c ; echo "======"; ./01-fork
/* (c) 2016-2017 Rahmat M. Samik-Ibrahim
 * http://rahmatm.samik-ibrahim.vlsm.org/
 * This is free software.
 */
#include <stdio.h>
#include <unistd.h>
#include <sys/types.h>
#include <sys/wait.h>

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

   printf("PID[%d] PPID[%d] (START:%s)\n", getpid(), getppid(), iAM);
   if (fork() > 0) {
      sleep(1);     /* LOOK THIS ************** */
      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);
}
======
PID[1428] PPID[1239] (START:PARENT)
PID[1429] PPID[1428] (ELSE:CHILD)
PID[1429] PPID[1428] (STOP:CHILD)
PID[1428] PPID[1239] (IFF0:PARENT)
PID[1428] PPID[1239] (STOP:PARENT)
>>>>> $
```

02-fork

```
>>>>> $ cat 02-fork.c ; echo "======" ; ./02-fork
/* (c) 2016-2017 Rahmat M. Samik-Ibrahim
 * http://rahmatm.samik-ibrahim.vlsm.org/
 * This is free software.
 */
#include <stdio.h>
#include <unistd.h>
#include <sys/types.h>
#include <sys/wait.h>

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

   printf("PID[%d] PPID[%d] (START:%s)\n", getpid(), getppid(), iAM);
   if (fork() > 0) {
      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);
      sleep(1);     /* LOOK THIS ************** */
   }
   printf("PID[%d] PPID[%d] (STOP:%s)\n", getpid(), getppid(), iAM);
}
======
PID[1541] PPID[1239] (START:PARENT)
PID[1541] PPID[1239] (IFF0:PARENT)
PID[1541] PPID[1239] (STOP:PARENT)
PID[1542] PPID[1541] (ELSE:CHILD)
>>>>> $ PID[1542] PPID[1] (STOP:CHILD)
>>>>> $
```

03-fork

```
>>>>> $ cat 03-fork.c ; echo "======"; ./03-fork
/* (c) 2016-2017 Rahmat M. Samik-Ibrahim
 * http://rahmatm.samik-ibrahim.vlsm.org/
 * This is free software.
 */
#include <stdio.h>
#include <unistd.h>
#include <sys/types.h>
#include <sys/wait.h>

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

   printf("PID[%d] PPID[%d] (START:%s)\n", getpid(), getppid(), iAM);
   if (fork() > 0) {
      wait(NULL);     /* LOOK THIS ************** */
      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);
}
======
PID[1590] PPID[1239] (START:PARENT)
PID[1591] PPID[1590] (ELSE:CHILD)
PID[1591] PPID[1590] (STOP:CHILD)
PID[1590] PPID[1239] (IFF0:PARENT)
PID[1590] PPID[1239] (STOP:PARENT)
>>>>> $
```

01-fork vs 02-fork vs 03-fork

```
>>>>> $ ./01-fork
PID[1726] PPID[1239] (START:PARENT)
PID[1727] PPID[1726] (ELSE:CHILD)
PID[1727] PPID[1726] (STOP:CHILD)
PID[1726] PPID[1239] (IFF0:PARENT)
PID[1726] PPID[1239] (STOP:PARENT)
>>>>> $ ./02-fork
PID[1728] PPID[1239] (START:PARENT)
PID[1728] PPID[1239] (IFF0:PARENT)
PID[1728] PPID[1239] (STOP:PARENT)
PID[1729] PPID[1728] (ELSE:CHILD)
>>>>> $ PID[1729] PPID[1] (STOP:CHILD)
>>>>> $ ./03-fork
PID[1730] PPID[1239] (START:PARENT)
PID[1731] PPID[1730] (ELSE:CHILD)
PID[1731] PPID[1730] (STOP:CHILD)
PID[1730] PPID[1239] (IFF0:PARENT)
PID[1730] PPID[1239] (STOP:PARENT)
>>>>> $
```

04-sleeping

```
#include <stdio.h>
#include <unistd.h>
void main(void) {
   int ii;
   printf("Sleeping 3 seconds: ");
   fflush(NULL);
   for (ii=0; ii < 3; ii++) {
      sleep(1);
      printf("x ");
      fflush(NULL);
   }
   printf("\nSleeping with no fflush(): ");
   for (ii=0; ii < 3; ii++) {
      sleep(1);
      printf("x ");
   }
   printf("\n");
}
Sleeping 3 seconds: x x x          (every 1 second)
Sleeping with no fflush(): x x x              (all)
```

05-fork

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

void main(void) {
   printf("Start:   PID[%d] PPID[%d]\n", getpid(), getppid());
   fflush(NULL);
   if (fork() == 0) {
      printf("Child:   00-fork >>> >>> >>> ");
      fflush(NULL);
      /* START BLOCK
         END   BLOCK */
      execlp("./00-fork", "00-fork", NULL);
   } else {
      wait(NULL);
      printf("Parent:  ");
   }
   printf(        "PID[%d] PPID[%d]  <<< <<< <<<\n", getpid(), getppid());
}
```

```
execlp ======================
Start:   PID[2534] PPID[1239]
Child:   00-fork >>> >>> >>> 00-fork:  PID[2535] PPID[2534]
Parent:  PID[2534] PPID[1239]  <<< <<< <<<

no execlp ===================
Start:   PID[2543] PPID[1239]
Child:   00-fork >>> >>> >>> PID[2544] PPID[2543]  <<< <<< <<<
Parent:  PID[2543] PPID[1239]  <<< <<< <<<
```

X

X

The End

- This is the end of the presentation.