

# CSGE602055 Operating Systems

## CSF2600505 Sistem Operasi

### Week 06: Concurrency: Processes & Threads

C. BinKadal

Sendirian Berhad

<https://os.vlsm.org/Slides/os06.pdf>

Always check for the latest revision!

REV392 30-Aug-2022

# OS222<sup>3</sup>): Operating Systems Schedule 2022 - 2

Week	Topic <sup>1)</sup>	OSC10 <sup>2)</sup>
Week 00	Overview (1), Assignment of Week 00	Ch. 1, 2
Week 01	Overview (2), Virtualization & Scripting	Ch. 1, 2, 18.
Week 02	Security, Protection, Privacy, & C-language.	Ch. 16, 17.
Week 03	File System & FUSE	Ch. 13, 14, 15.
Week 04	Addressing, Shared Lib, & Pointer	Ch. 9.
Week 05	Virtual Memory	Ch. 10.
Week 06	Concurrency: Processes & Threads	Ch. 3, 4.
Week 07	Synchronization & Deadlock	Ch. 6, 7, 8.
Week 08	Scheduling + W06/W07	Ch. 5.
Week 09	Storage, Firmware, Bootloader, & Systemd	Ch. 11.
Week 10	I/O & Programming	Ch. 12.

<sup>1)</sup> For schedule, see <https://os.vlsm.org/#idx02>

<sup>2)</sup> Silberschatz et. al.: **Operating System Concepts**, 10<sup>th</sup> Edition, 2018.

<sup>3)</sup> This information will be on **EVERY** page two (2) of this course material.

# STARTING POINT — <https://os.vlsm.org/>

- ☐ **Text Book** — Any recent/decent OS book. Eg. (**OSC10**) Silberschatz et. al.: **Operating System Concepts**, 10<sup>th</sup> Edition, 2018. (See <https://www.os-book.com/OS10/>).
- ☐ **Resources** (<https://os.vlsm.org/#idx03>)
  - ☐ **SCELE OS222** — <https://scele.cs.ui.ac.id/course/view.php?id=3398>.  
The enrollment key is **XXX**.
  - ☐ **Download Slides and Demos from GitHub.com** — (<https://github.com/os2xx/os/>)  
[os00.pdf \(W00\)](#), [os01.pdf \(W01\)](#), [os02.pdf \(W02\)](#), [os03.pdf \(W03\)](#), [os04.pdf \(W04\)](#), [os05.pdf \(W05\)](#),  
[os06.pdf \(W06\)](#), [os07.pdf \(W07\)](#), [os08.pdf \(W08\)](#), [os09.pdf \(W09\)](#), [os10.pdf \(W10\)](#).
  - ☐ **Problems**  
[195.pdf \(W00\)](#), [196.pdf \(W01\)](#), [197.pdf \(W02\)](#), [198.pdf \(W03\)](#), [199.pdf \(W04\)](#), [200.pdf \(W05\)](#),  
[201.pdf \(W06\)](#), [202.pdf \(W07\)](#), [203.pdf \(W08\)](#), [204.pdf \(W09\)](#), [205.pdf \(W10\)](#).
  - ☐ **LFS** — <http://www.linuxfromscratch.org/lfs/view/stable/>
  - ☐ **OSP4DISS** — <https://osp4diss.vlsm.org/>
  - ☐ **This is How Me Do It!** — <https://doit.vlsm.org/001.html>
    - ☐ PS: "Me" rhymes better than "I", duh!

# Agenda

- 1 Start
- 2 OS222 Schedule
- 3 Agenda
- 4 Week 06
- 5 OSC10 (Silberschatz) Chapter 3: Processes and Chapter 4: Threads & Concurrency
- 6 Week 06
- 7 Process Map
- 8 Process State
- 9 Makefile
- 10 00-show-pid
- 11 01-fork
- 12 02-fork
- 13 03-fork
- 14 01-fork vs 02-fork vs 03-fork
- 15 04-sleep

## Agenda (2)

- 16 05-fork
- 17 06-fork
- 18 07-exec1p
- 19 08-fork
- 20 09-fork
- 21 10-fork
- 22 11-fork
- 23 12-fork
- 24 13-uas161
- 25 14-uas162
- 26 15-uas171
- 27 16-uas172
- 28 Assignment Week06

# Week 06 Concurrency: Topics<sup>1</sup>

- States and state diagrams
- Structures (ready list, process control blocks, and so forth)
- Dispatching and context switching
- The role of interrupts
- Managing atomic access to OS objects
- Implementing synchronization primitives
- Multiprocessor issues (spin-locks, reentrancy)

---

<sup>1</sup>Source: ACM IEEE CS Curricula 2013

## Week 06 Concurrency: Learning Outcomes (1)<sup>1</sup>

- Describe the need for concurrency within the framework of an operating system. [Familiarity]
- Demonstrate the potential run-time problems arising from the concurrent operation of many separate tasks. [Usage]
- Summarize the range of mechanisms that can be employed at the operating system level to realize concurrent systems and describe the benefits of each. [Familiarity]
- Explain the different states that a task may pass through and the data structures needed to support the management of many tasks. [Familiarity]

---

<sup>1</sup>Source: ACM IEEE CS Curricula 2013

## Week 06 Concurrency: Learning Outcomes (2)<sup>1</sup>

- Summarize techniques for achieving synchronization in an operating system (e.g., describe how to implement a semaphore using OS primitives). [Familiarity]
- Describe reasons for using interrupts, dispatching, and context switching to support concurrency in an operating system. [Familiarity]
- Create state and transition diagrams for simple problem domains. [Usage]

---

<sup>1</sup>Source: ACM IEEE CS Curricula 2013

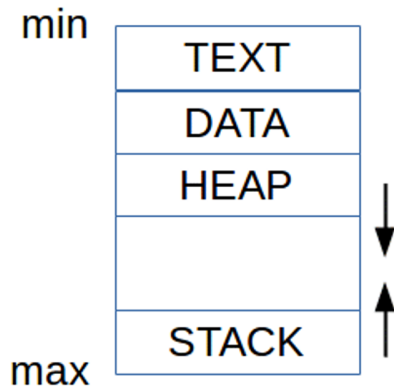


- Chapter 3: Processes
  - Process Concept
  - Process Scheduling
  - Operations on Processes
  - Interprocess Communication
  - IPC in Shared-Memory Systems
  - IPC in Message-Passing Systems
  - Examples of IPC Systems
  - Communication in Client-Server Systems
- Chapter 4: Threads & Concurrency
  - Overview
  - Multicore Programming
  - Multithreading Models
  - Thread Libraries
  - Implicit Threading
  - Threading Issues
  - Operating System Examples

# Week 06: Concurrency: Processes & Threads

- Reference: (OSC10-ch03 OSC10-ch04 demo-w06)
- Process Concept
  - Program (passive)  $\leftrightarrow$  Process (active)
  - Process in Memory: | *Stack*  $\cdots$  *Heap* | *Data* | *Text* |
  - Process State: | *running* | *waiting* | *ready* |
  - Process Control Block (PCB)
    - /proc/, Process State, Program Counter, Registers, Management Information.
- Process Creation
  - PID: Process Identifier (uniq)
  - The Parent Process forms a tree of Children Processes
  - `fork()`, new process system call (clone)
  - `exec1p()`, replaces the clone with a new program.
- Process Termination
  - `wait()`, until the child process is terminated.
- PCB (Context) Switch

## A PROCESS IN MEMORY



(c) 2017 VauLSMorg

Figure: A Process in (**logical**) Memory

# Process Map (2)

```
/*
 * Copyright (C) 2021 Rahmat M. Samik-Ibrahim
 * START: Sat 03 Apr 2021 06:20:43 WIB
 */

#include <stdio.h>
#include <stdlib.h>

typedef void* AnyAddrPtr;
typedef char* ChrPtr;
typedef char Chr;

Chr    aGlobalArray[16];
ChrPtr aGlobalCharacter1;
ChrPtr aGlobalCharacter2;
ChrPtr aGlobalCharacterPointer=aGlobalArray;

void printMyAddress (AnyAddrPtr address, ChrPtr message) {
    printf("[%p] %s\n", address, message);
}

int main(void) {
    ChrPtr aHeapCharacterPointer=malloc(16);
    Chr    aLocalArray[16];
    ChrPtr aLocalCharacterPointer=aGlobalArray;
    ChrPtr aLocalCharacter1;
    ChrPtr aLocalCharacter2;

    // ...
}
```

# Process Map (3)

[0x55559fcf9169]	printMyAddress	(function, TEXT)
[0x55559fcf919c]	main	(function, TEXT)
[0x55559fcfc010]	aGlobalCharacterPointer	(global variable, DATA)
[0x55559fcfc030]	aGlobalCharacter1	(global variable, DATA)
[0x55559fcfc040]	aGlobalArray	(global variable, DATA)
[0x55559fcfc050]	aGlobalCharacter2	(global variable, DATA)
[0x5555a0d192a0]	aHeapCharacterPointer	(HEAP)
[0x7f9377bc9e10]	printf	(library, SHARED)
[0x7f9377c02260]	malloc	(library, SHARED)
[0x7fff8caa0010]	aHeapCharacterPointer	(Pointer Variable, STACK)
[0x7ffd98ce1a10]	aLocalCharacterPointer	(local variable, STACK)
[0x7ffd98ce1a18]	aLocalCharacter1	(local variable, STACK)
[0x7ffd98ce1a20]	aLocalCharacter2	(local variable, STACK)
[0x7ffd98ce1a30]	aLocalArray	(local variable, STACK)

# Process State

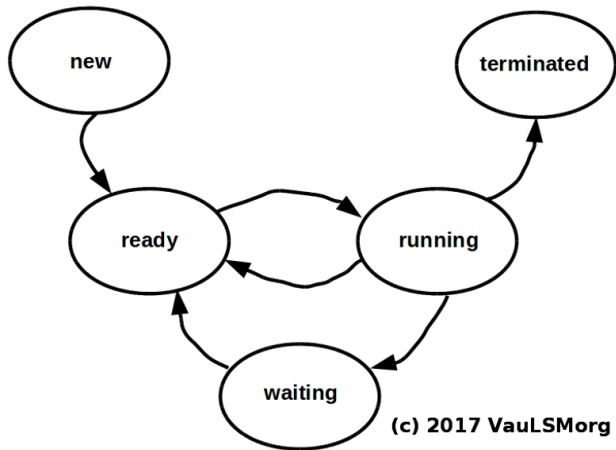


Figure: A Process State

# Process Scheduling

- Scheduling Queue
- Schedulers
  - Long Term (non VM) vs Short Term (CPU)
  - (I/O vs CPU) Bound Processes
- Context Switch
- I/O Queue Scheduling
- 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.

# Inter-Process Communication (IPC)

- Independent vs Cooperating Processes.
  - Cooperation: Information Sharing, Computational Speedup, Modularity, Convenience.
- Shared Memory vs Message Passing.
  - Message Passing: Direct vs Indirect Communication
- Client-Server Systems
  - Sockets
  - RPC: Remote Procedure Calls
  - Pipes



- Single vs Multithreaded Process
  - MultiT Benefits: Responsiveness, Resource Sharing, Economy, Scalability
- Multicore Programming
  - Concurrency vs. Parallelism
- Multithreading Models (Kernel vs User Thread)
  - Many to One
  - One to One
  - Many to Many
  - Multilevel Models
- Threading Issues
  - Parallelism on a multi-core system.
- Pthreads

# Makefile

```
CC='gcc'
CFLAGS='-std=c99'

P00=00-show-pid
...
P15=15-uas171
P16=16-uas172

EXECS= \
    $(P00) \
    $(P01) \
    ....
    $(P15) \
    $(P16) \

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)

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

# 00-show-pid

```
/*  
 * (c) 2016-2020 Rahmat M. Samik-Ibrahim  
 * https://rahmatm.samik-ibrahim.vlsm.org/  
 * This is free software.  
 * REV07 Tue Mar 24 12:06:10 WIB 2020  
 * START Mon Oct 24 09:42:05 WIB 2016  
 */  
  
#include <stdio.h>  
#include <unistd.h>  
#include <sys/types.h>  
  
void main(void) {  
    printf("  [[[ This is 00-show-pid: PID[%d] PPID[%d] ]]]\n",  
           getpid(), getppid());  
}  
  
>>>> $ ./00-show-pid
```

# 01-fork

```
>>>> $ cat 01-fork.c ; echo "=====" ; ./01-fork
/* (c) 2016-2017 Rahmat M. Samik-Ibrahim
 * https://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] (IFFO:%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[5784] PPID[1350] (START:PARENT)
PID[5785] PPID[5784] (ELSE:CHILD)
PID[5785] PPID[5784] (STOP:CHILD)
PID[5784] PPID[1350] (IFFO:PARENT)
PID[5784] PPID[1350] (STOP:PARENT)
>>>> $
```

## 02-fork

```
>>>> $ cat 02-fork.c ; echo "=====" ; ./02-fork
/* (c) 2016-2017 Rahmat M. Samik-Ibrahim
 * https://rahmatm.samik-ibrahim.uism.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] (IFFO:%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[5792] PPID[1350] (START:PARENT)
PID[5792] PPID[1350] (IFFO:PARENT)
PID[5792] PPID[1350] (STOP:PARENT)
PID[5793] PPID[5792] (ELSE:CHILD)
>>>> $ PID[5793] PPID[1] (STOP:CHILD)
>>>> $
```

# 03-fork

```
>>>> $ cat 03-fork.c ; echo "=====" ; ./03-fork
/* (c) 2016-2017 Rahmat M. Samik-Ibrahim
 * https://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] (IFFO:%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[5799] PPID[1350] (START:PARENT)
PID[5800] PPID[5799] (ELSE:CHILD)
PID[5800] PPID[5799] (STOP:CHILD)
PID[5799] PPID[1350] (IFFO:PARENT)
PID[5799] PPID[1350] (STOP:PARENT)
>>>> $
```

# 01-fork vs 02-fork vs 03-fork

```
>>>> $ ./01-fork
PID[5803] PPID[1350] (START:PARENT)
PID[5804] PPID[5803] (ELSE:CHILD)
PID[5804] PPID[5803] (STOP:CHILD)
PID[5803] PPID[1350] (IFF0:PARENT)
PID[5803] PPID[1350] (STOP:PARENT)
>>>> $ ./02-fork
PID[5805] PPID[1350] (START:PARENT)
PID[5805] PPID[1350] (IFF0:PARENT)
PID[5805] PPID[1350] (STOP:PARENT)
PID[5806] PPID[5805] (ELSE:CHILD)
>>>> $ PID[5806] PPID[1] (STOP:CHILD)
>>>> $ ./03-fork
PID[5807] PPID[1350] (START:PARENT)
PID[5808] PPID[5807] (ELSE:CHILD)
PID[5808] PPID[5807] (STOP:CHILD)
PID[5807] PPID[1350] (IFF0:PARENT)
PID[5807] PPID[1350] (STOP:PARENT)
>>>> $
```

## 04-sleep

```
#include <stdio.h>
#include <unistd.h>
void main(void) {
    int ii;
    printf("Sleeping 3s with fflush(): ");
    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 3s with fflush(): x x x
Sleeping with no fflush(): x x x
```



```

#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) {
        /* START BLOCK
        execlp("./00-fork", "00-fork", NULL);
        END BLOCK */
        printf("Child:          ");
    } else {
        wait(NULL);
        printf("Parent:          ");
    }
    printf("          PID[%d] PPID[%d] <<< <<< <<<\n", getpid(), getppid());
}

```

no execlp =====

```

Start:          PID[6040] PPID[1350]
Child:          PID[6041] PPID[6040] <<< <<< <<<
Parent:         PID[6040] PPID[1350] <<< <<< <<<

```

# 05b-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) {
        /* START BLOCK
           END   BLOCK */
        execlp("./00-fork", "00-fork", NULL);
        printf("Child:          ");
    } else {
        wait(NULL);
        printf("Parent:          ");
    }
    printf("          PID[%d] PPID[%d] <<< <<< <<<\n", getpid(), getppid());
}

execlp =====
Start:          PID[6007] PPID[1350]
[[[ This is 00-show-pid: PID[6008] PPID[6007] ]]]
Parent:          PID[6007] PPID[1350] <<< <<< <<<
```

# 06a-fork

```
#include <sys/types.h>
#include <sys/wait.h>
#include <stdio.h>
#include <stdlib.h>
#include <unistd.h>
/***** main ***/
void main(void) {
    pid_t val1, val2, val3;
    val3 = val2 = val1 = 1000;
    printf("PID==%4d ==== \n", getpid());
    fflush(NULL);
    val1 = fork();
    wait(NULL);
    val2 = fork();
    wait(NULL);
    val3 = fork();
    wait(NULL);
    /* ***** START BLOCK *
       ***** END** BLOCK */
    printf("VAL1=%4d VAL2=%4d VAL3=%4d\n", val1, val2, val3);
}

=====
PID==[13965] ==== 
VAL1=[01000] VAL2=[01000] VAL3=[01000]
```

## 06b-fork

```
#include <sys/types.h>
#include <sys/wait.h>
#include <stdio.h>
#include <stdlib.h>
#include <unistd.h>
/***** main ***/
void main(void) {
    pid_t val1, val2, val3;
    val3 = val2 = val1 = 1000;
    printf("PID==%4d ==== \n", getpid());
    fflush(NULL);
    val1 = fork();
    wait(NULL);
    /* ***** START BLOCK */
    val2 = fork();
    wait(NULL);
    val3 = fork();
    wait(NULL);
    /* ***** END** BLOCK */
    printf("VAL1=%4d VAL2=%4d VAL3=%4d\n", val1, val2, val3);
}

=====
PID==[13969] ==== 
VAL1=[00000] VAL2=[01000] VAL3=[01000]
VAL1=[13970] VAL2=[01000] VAL3=[01000]
```

```
#include <sys/types.h>
#include <sys/wait.h>
#include <stdio.h>
#include <stdlib.h>
#include <unistd.h>
/***** main ** */
void main(void) {
    pid_t val1, val2, val3;
    val3 = val2 = val1 = 1000;
    printf("PID==%4d ==== \n", getpid());
    fflush(NULL);
    val1 = fork();
    wait(NULL);
    val2 = fork();
    wait(NULL);
    /* ***** START BLOCK */
    val3 = fork();
    wait(NULL);
    ***** END** BLOCK */
    printf("VAL1=%4d VAL2=%4d VAL3=%4d\n", val1, val2, val3);
}

=====
PID==[13971] ==== 
VAL1=[00000] VAL2=[00000] VAL3=[01000]
VAL1=[00000] VAL2=[13973] VAL3=[01000]
VAL1=[13972] VAL2=[00000] VAL3=[01000]
VAL1=[13972] VAL2=[13974] VAL3=[01000]
```

# 06d-fork

```
#include <sys/types.h>
#include <sys/wait.h>
#include <stdio.h>
#include <stdlib.h>
#include <unistd.h>
/***** main ** */
void main(void) {
    pid_t val1, val2, val3;
    val3 = val2 = val1 = 1000;
    printf("PID==%4d ==== \n", getpid());
    fflush(NULL);
    val1 = fork();
    wait(NULL);
    val2 = fork();
    wait(NULL);
    val3 = fork();
    wait(NULL);

    /* ***** START BLOCK * ***** END** BLOCK */
    printf("VAL1=%4d VAL2=%4d VAL3=%4d\n", val1, val2, val3);
}

=====
PID==[13976] ==== 
VAL1=[00000] VAL2=[00000] VAL3=[00000]
VAL1=[00000] VAL2=[00000] VAL3=[13979]
VAL1=[00000] VAL2=[13978] VAL3=[00000]
VAL1=[00000] VAL2=[13978] VAL3=[13980]
VAL1=[13977] VAL2=[00000] VAL3=[00000]
VAL1=[13977] VAL2=[00000] VAL3=[13982]
VAL1=[13977] VAL2=[13981] VAL3=[00000]
VAL1=[13977] VAL2=[13981] VAL3=[13983]
```

# 07-exec1p

```
>>>> $ cat 07-exec1p.c
/* (c) 2019-2020 Rahmat M. Samik-Ibrahim
 * https://rahmatm.samik-ibrahim.vlsm.org/
 * This is free software.
 * REV01 Tue Mar 24 16:29:50 WIB 2020
 * START Mon Dec 9 16:28:36 WIB 2019
 */
#include <stdio.h>
#include <sys/types.h>
#include <unistd.h>
void main(int argc, char* argv[]) {
    printf("START %11s PID[%d]\n", argv[0], getpid());
    if(argc == 1) {
        exec1p(argv[0], "EXECLP", "Whatever", NULL);
    } else {
        printf("ELSE %11s PID[%d]\n", argv[1], getpid());
    }
    printf("END %11s PID[%d]\n", argv[0], getpid());
}

$ ./07-exec1p
START ./07-exec1p PID[14172]
START EXECLP PID[14172]
ELSE Whatever PID[14172]
END EXECLP PID[14172]
$ ./07-exec1p XYZZYPLUGH
START ./07-exec1p PID[14174]
ELSE XYZZYPLUGH PID[14174]
END ./07-exec1p PID[14174]
$
```

# 08-fork

```
/* (c) 2005-2017 Rahmat M. Samik-Ibrahim https://rahmatm.samik-ibrahim.vlsm.org/ This is free software.
 * REV02 Thu Oct 26 12:27:30 WIB 2017
 * START 2005
 */
#include <sys/types.h>
#include <sys/wait.h>
#include <stdio.h>
#include <stdlib.h>
#include <unistd.h>
void main(void) {
    int ii=0;
    if (fork() == 0) ii++;
    wait(NULL);
    if (fork() == 0) ii++;
    wait(NULL);
    if (fork() == 0) ii++;
    wait(NULL);
    printf ("Result = %d \n",ii);
    exit(0);
}
=====
Result = 3
Result = 2
Result = 2
Result = 1
Result = 2
Result = 1
Result = 1
Result = 0
>>>> $
```



# 09-fork

```
/*
 * (c) 2015-2017 Rahmat M. Samik-Ibrahim https://rahmatm.samik-ibrahim.vlsm.org/
 * REV03 Mon Oct 30 11:04:10 WIB 2017
 * REV00 Mon Oct 24 10:43:00 WIB 2016
 * START 2015
 */
#include <stdio.h>
#include <sys/types.h>
#include <sys/wait.h>
#include <unistd.h>

void main(void) {
    int value;

    value=fork();
    wait(NULL);
    printf("I am PID[%4d] -- The fork() return value is: %4d\n", getpid(), value);

    value=fork();
    wait(NULL);
    printf("I am PID[%4d] -- The fork() return value is: %4d\n", getpid(), value);
}

=====
I am PID[6225] -- The fork() return value is:    0)
I am PID[6226] -- The fork() return value is:    0)
I am PID[6225] -- The fork() return value is: 6226)
I am PID[6224] -- The fork() return value is: 6225)
I am PID[6227] -- The fork() return value is:    0)
I am PID[6224] -- The fork() return value is: 6227)
>>>> $
```

# 10-fork

```
/* (c) 2016-2017 Rahmat M. Samik-Ibrahim https://rahmatm.samik-ibrahim.vlsm.org/ This is free software.
```

```
 * REV02 Mon Oct 30 20:25:44 WIB 2017
```

```
 */
```

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

```
void procStatus(int level) {
    printf("L%d: PID[%d] (PPID[%d])\n", level, getpid(), getppid());
    fflush(NULL);
}
```

```
int addLevelAndFork(int level) {
    if (fork() == 0) level++;
    wait(NULL);
    return level;
}
```

```
void main(void) {
    int level = 0;
    procStatus(level);
    level = addLevelAndFork(level);
    procStatus(level);
}
```

```
=====
```

```
L0: PID[7540] (PPID[1350])
```

```
L1: PID[7541] (PPID[7540])
```

```
L0: PID[7540] (PPID[1350])
```

# 11-fork

```
/* (c) 2016-2017 Rahmat M. Samik-Ibrahim https://rahmatm.samik-ibrahim.vlsm.org/ This is free software.
 * REV02 Mon Oct 30 20:27:24 WIB 2017
 * START Mon Oct 24 09:42:05 WIB 2016
 */

#define LOOP 3
#include <stdio.h>
#include <sys/types.h>
#include <sys/wait.h>
#include <unistd.h>

void procStatus(int level) {
    printf("L%d: PID[%d] (PPID[%d])\n", level, getpid(), getppid());
    fflush(NULL);
}

int addLevelAndFork(int level) {
    if (fork() == 0) level++;
    wait(NULL);
    return level;
}

void main(void) {
    int ii, level = 0;
    procStatus(level);
    for (ii=0;ii<LOOP;ii++) {
        level = addLevelAndFork(level);
        procStatus(level);
    }
}
```

# 11-fork (2)

```
L0: PID[7548] (PPID[1350])
L1: PID[7549] (PPID[7548])
L2: PID[7550] (PPID[7549])
L3: PID[7551] (PPID[7550])
L2: PID[7550] (PPID[7549])
L1: PID[7549] (PPID[7548])
L2: PID[7552] (PPID[7549])
L1: PID[7549] (PPID[7548])
L0: PID[7548] (PPID[1350])
L1: PID[7553] (PPID[7548])
L2: PID[7554] (PPID[7553])
L1: PID[7553] (PPID[7548])
L0: PID[7548] (PPID[1350])
L1: PID[7555] (PPID[7548])
L0: PID[7548] (PPID[1350])
```

# 12-fork

```
#include <stdio.h>
#include <unistd.h>
#include <sys/types.h>
#include <sys/wait.h>
void waitAndPrintPID(void) {
    wait(NULL);
    printf("PID: %d\n", getpid());
    fflush(NULL);
}
void main(int argc, char *argv[]) {
    int rc, status;
    waitAndPrintPID();
    rc = fork();
    waitAndPrintPID();
    if (rc == 0) {
        fork();
        waitAndPrintPID();
        execlp("./00-fork", "00-fork", NULL);
    }
    waitAndPrintPID();
}
=====
PID: 7614
PID: 7615
PID: 7616
[[[ This is 00-fork: PID[7616] PPID[7615] ]]]
PID: 7615
[[[ This is 00-fork: PID[7615] PPID[7614] ]]]
PID: 7614
PID: 7614
```

```

/*
 * Copyright (C) 2015-2020 Rahmat M. Samik-Ibrahim http://rahmatm.samik-ibrahim.vlsm.org/ This program is free script/software.
 * REV10 Tue Mar 24 16:38:29 WIB 2020
 * START Xxx Xxx XX XX:XX:XX XXX XXXX
 */

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

void main(void) {
    pid_t pid1, pid2, pid3;

    pid1 = pid2 = pid3 = getpid();
    printf(" 2016    2015    Lainnya\n=====\\n");
    printf("[%5.5d] [%5.5d] [%5.5d]\\n", pid1, pid2, pid3);
    fork();
    pid1 = getpid();
    wait(NULL);
    pid2 = getpid();
    if(!fork()) {
        pid2 = getpid();
        fork();
    }
    pid3 = getpid();
    wait(NULL);
    printf("[%5.5d] [%5.5d] [%5.5d]\\n", pid1, pid2, pid3);
}

```

```
/*  
# INFO: UTS 2016-1 (midterm)  
*/
```

```
$ ./13-uas161  
2016    2015    Lainnya
```

```
=====
```

```
[14492] [14492] [14492]  
[14493] [14494] [14495]  
[14493] [14494] [14494]  
[14493] [14493] [14493]  
[14492] [14496] [14497]  
[14492] [14496] [14496]  
[14492] [14492] [14492]
```

```

/* Copyright (C) 2016-2020 Rahmat M. Samik-Ibrahim http://rahmatm.samik-ibrahim.vlsm.org/
 * This program is free script/software.
 * REV08 Tue Mar 24 16:40:28 WIB 2020
 * START Sun Dec 04 00:00:00 WIB 2016
 * wait()      = suspends until its child terminates.
 * fflush()    = flushes the user-space buffers.
 * getppid()   = get parent PID
 * ASSUME pid >= 1000 ES pid > ppid **
 */

```

```

#include <stdio.h>
#include <sys/types.h>
#include <unistd.h>
#include <sys/wait.h>
#define NN 2

void main(void) {
    int ii, rPID, rPPID, id1000=getpid();
    for (ii=1; ii<=NN; ii++) {
        fork();
        wait(NULL);
        rPID = getpid()-id1000+1000; /* "relative" */
        rPPID=getppid()-id1000+1000; /* "relative" */
        if (rPPID < 1000 || rPPID > rPID) rPPID=999;
        printf("Loop [%d] - rPID[%d] - rPPID[%4d]\n", ii, rPID, rPPID);
        fflush(NULL);
    }
}

```



```
/*  
# INFO: UTS 2016-2 (midterm)  
*/
```

```
$ ./14-uas162  
Loop [1] - rPID[1001] - rPPID[1000]  
Loop [2] - rPID[1002] - rPPID[1001]  
Loop [2] - rPID[1001] - rPPID[1000]  
Loop [1] - rPID[1000] - rPPID[ 999]  
Loop [2] - rPID[1003] - rPPID[1000]  
Loop [2] - rPID[1000] - rPPID[ 999]
```

```

/* Copyright (C) 2005-2020 Rahmat M. Samik-Ibrahim http://rahmatm.samik-ibrahim.vlsm.org/ This program is free script/software.
 * REV00 Wed May 3 17:07:09 WIB 2017
 * START 2005
 * fflush(NULL): flushes all open output streams
 * fork():      creates a new process by cloning
 * getpid():     get PID (Process ID)
 * wait(NULL):   wait until the child is terminated
 */

```

```

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

```

```

void main(void) {
    int firstPID = (int) getpid();
    int RelPID;

    fork();
    wait(NULL);
    fork();
    wait(NULL);
    fork();
    wait(NULL);

    RelPID=(int)getpid()-firstPID+1000;
    printf("RelPID: %d\n", RelPID);
    fflush(NULL);
}

```

```
/*  
# INFO: UTS 2017-1 (midterm)  
*/
```

```
$ ./15-uas171
```

```
RelPID: 1003
```

```
RelPID: 1002
```

```
RelPID: 1004
```

```
RelPID: 1001
```

```
RelPID: 1006
```

```
RelPID: 1005
```

```
RelPID: 1007
```

```
RelPID: 1000
```

```
$
```

```

/*
 * (c) 2017-2020 Rahmat M. Samik-Ibrahim
 * http://rahmatm.samik-ibrahim.vlsm.org/
 * This is free software.
 * REV03 Tue Mar 24 16:42:16 WIB 2020
 * REV02 Mon Dec 11 17:46:01 WIB 2017
 * START Sun Dec 3 18:00:08 WIB 2017
 */

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

#define LOOP 3
#define OFFSET 1000

void main(void) {
    int basePID = getpid() - OFFSET;

    for (int ii=0; ii < LOOP; ii++) {
        if(!fork()) {
            printf("PID[%d]-PPID[%d]\n",
                getpid() - basePID,
                getppid() - basePID);
            fflush(NULL);
        }
        wait(NULL);
    }
}

```

```
/*  
# INFO: UTS 2017-2 (midterm)  
*/
```

```
$ ./16-uas172  
PID[1001]-PPID[1000]  
PID[1002]-PPID[1001]  
PID[1003]-PPID[1002]  
PID[1004]-PPID[1001]  
PID[1005]-PPID[1000]  
PID[1006]-PPID[1005]  
PID[1007]-PPID[1000]  
$
```

# mylib.h (1)

```
/*
 * Copyright (C) 2021-2021 Rahmat M. Samik-Ibrahim
 * http://rahmatm.samik-ibrahim.vlsm.org/
 * This program is free script/software. This program is distributed in the
 * hope that it will be useful, but WITHOUT ANY WARRANTY; without even the
 * implied warranty of MERCHANTABILITY or FITNESS FOR A PARTICULAR PURPOSE.
 * REV08: Sun 04 Apr 07:28:09 WIB 2021
 * REV07: Sun 04 Apr 00:11:43 WIB 2021
 * REV06: Sat 03 Apr 11:00:46 WIB 2021
 * REV05: Tue 30 Mar 14:55:36 WIB 2021
 * REV04: Tue 30 Mar 10:35:13 WIB 2021
 * START: Mon 22 Mar 16:14:36 WIB 2021
 *
# INFO: mylib.h
*/

#define TOKEN            "OS212W06"
#define WEEKFILE         "WEEK06-MEMORY-SHARE.bin"
#define FORKS            4

#define BUFFERSIZE       256
#define SSIZE            4
#define STAMPSIZE        11
#define CHMOD            0666
#define CMDSHA1 "echo %s | shasum | cut -c1-4 | tr '[:lower:]' '[:upper:]' "
#define MYFLAGS          O_CREAT|O_RDWR
#define MYPROTECTION      PROT_READ|PROT_WRITE
#define MYVISIBILITY      MAP_SHARED
```

# mylib.h (2)

```
#include <fcntl.h>
#include <stdio.h>
#include <stdlib.h>
#include <string.h>
#include <sys/mman.h>
#include <sys/stat.h>
#include <sys/types.h>
#include <sys/wait.h>
#include <time.h>
#include <unistd.h>

typedef          char  Chr;
typedef          char* ChrPtr;
typedef unsigned char  uChr;
typedef unsigned char* uChrPtr;
typedef struct {
    Chr counter;
    Chr blank;
    Chr stamp[FORKS][BUFFERSIZE];
    Chr end;
    Chr zero;
} memStruct;
typedef memStruct* memStructPtr;

void          chktoken          (uChrPtr result, uChrPtr token);
memStructPtr createShareMemory(memStructPtr mymap, int memorySize, ChrPtr memoryName);
void          getTimeStamp      (uChrPtr timeStamp);
void          mySHA1             (uChrPtr output, uChrPtr input, int length);
void          pickToken          (uChrPtr result, uChrPtr token);
void          verifyToken        (uChrPtr result, uChrPtr token, uChrPtr input);
```

# mylib.c (1)

```
/*
 * Copyright (C) 2021-2021 Rahmat M. Samik-Ibrahim
 * http://rahmatm.samik-ibrahim.vlsm.org/
 * This program is free script/software. This program is distributed in the
 * hope that it will be useful, but WITHOUT ANY WARRANTY; without even the
 * implied warranty of MERCHANTABILITY or FITNESS FOR A PARTICULAR PURPOSE.
 * REV08: Sun 04 Apr 07:25:24 WIB 2021
 * REV07: Sun 04 Apr 00:11:43 WIB 2021
 * REV04: Tue 30 Mar 10:35:13 WIB 2021
 * START: Mon 22 Mar 16:14:36 WIB 2021
 *
# INFO: mylib.c
*/

#include "mylib.h"

void mySHA1(uChrPtr output, uChrPtr input, int length) {
    Chr      cmd[BUFFERSIZE];
    sprintf(cmd, CMDSHA1, input);
    FILE* filePtr = popen(cmd, "r");
    fgets(output, length+1, filePtr);
    output[length]=0;
    pclose(filePtr);
}

void getTimeStamp(uChrPtr timeStamp) {
    time_t tt      = time(NULL);
    struct tm tm = *localtime(&tt);
    sprintf(timeStamp, "%2.2d%2.2d", tm.tm_min, tm.tm_sec);
}
```



```
void chktoken (uChrPtr result, uChrPtr token) {
    uChr timeStamp[] = "MMSS";
    getTimeStamp(timeStamp);
    uChr input [BUFFERSIZE];
    strcpy(input,timeStamp);
    uChrPtr user=getenv("USER");
    strcat(input,user);
    strcat(input,token);
    uChr  output [BUFFERSIZE];
    mySHA1(output, input, SSIZE);
    sprintf(result, "%s %s-%s", user, timeStamp, output);
}

void verifyToken(uChrPtr result, uChrPtr token, uChrPtr input) {
    uChr  tmpStr1[BUFFERSIZE];
    uChr  tmpStr2[BUFFERSIZE];
    strcpy(tmpStr1,input);
    uChrPtr user=strtok(tmpStr1," ");
    uChrPtr timeStamp=strtok(NULL,"-");
    strcpy(tmpStr2,timeStamp);
    strcat(tmpStr2,user);
    strcat(tmpStr2,token);
    uChr  output [BUFFERSIZE];
    mySHA1(output, tmpStr2, SSIZE);
    uChrPtr tmpStr3=strtok(NULL,"-");
    if (strcmp(output, tmpStr3) == 0 ) sprintf(result, "Verified");
    else sprintf(result, "Error");
}
```

```
void pickToken (uChrPtr result, uChrPtr token) {
    uChr  tmpStr1[BUFFERSIZE];
    strcpy(tmpStr1,token);
    strtok(tmpStr1," ");
    strcpy(result, strtok(NULL," "));
}

memStructPtr createShareMemory(memStructPtr mymap, int memorySize, ChrPtr memoryName) {
    int    fd    = open(memoryName, MYFLAGS, CHMOD);
    fchmod (fd, CHMOD);
    ftruncate(fd, memorySize);
    mymap = mmap(NULL, memorySize, MYPROTECTION, MYVISIBILITY, fd, 0);
    close(fd);
    return mymap;
}
```

# chktoken.c (1)

```
/*
 * Copyright (C) 2021-2021 Rahmat M. Samik-Ibrahim
 * http://rahmatm.samik-ibrahim.vlsm.org/
 * This program is free script/software. This program is distributed in the
 * hope that it will be useful, but WITHOUT ANY WARRANTY; without even the
 * implied warranty of MERCHANTABILITY or FITNESS FOR A PARTICULAR PURPOSE.
# INFO: chktoken TOKEN
 * REVO2 Sun 04 Apr 2021 08:05:57 WIB
 * REVO1 Sun 04 Apr 2021 00:11:27 WIB
 * START Sat 03 Apr 2021 15:10:28 WIB
 */

#include "mylib.h"

int main(int argc, ChrPtr argv[]) {
    if (argc < 2) return -1;
    uChr    result1[BUFFERSIZE];
    chktoken (result1, argv[1]);
    printf("%s\n", result1);
}
```

# verifyToken.c (1)

```
/*
 * Copyright (C) 2021-2021 Rahmat M. Samik-Ibrahim
 * http://rahmatm.samik-ibrahim.vlsm.org/
 * This program is free script/software. This program is distributed in the
 * hope that it will be useful, but WITHOUT ANY WARRANTY; without even the
 * implied warranty of MERCHANTABILITY or FITNESS FOR A PARTICULAR PURPOSE.
# INFO: TOP (Table of Processes)
 * REVO2 Sun 04 Apr 2021 07:24:22 WIB
 * REVO1 Sun 04 Apr 2021 00:11:27 WIB
 * START Sat 03 Apr 2021 15:10:28 WIB
 */

#include "mylib.h"

int main(int argc, ChrPtr argv[]) {
    if (argc < 4) return -1;
    uChr    result1[BUFFERSIZE];
    uChr    result2[BUFFERSIZE];
    strcpy(result1,argv[2]);
    strcat(result1," ");
    strcat(result1,argv[3]);
    verifyToken(result2, argv[1], result1);
    printf("%s\n", result2);
}
```

# myfork.c (1)

```
/*
 * Copyright (C) 2021-2021 Rahmat M. Samik-Ibrahim
 * http://rahmatm.samik-ibrahim.vlsm.org/
 * This program is free script/software. This program is distributed in the
 * hope that it will be useful, but WITHOUT ANY WARRANTY; without even the
 * implied warranty of MERCHANTABILITY or FITNESS FOR A PARTICULAR PURPOSE.
# INFO: myfork00
 * START Sun 04 Apr 2021 11:00:01 AM WIB
 */

#include "mylib.h"

int main(void) {
    memStructPtr mymap = createShareMemory(mymap, sizeof(memStruct), WEEKFILE);
    mymap->counter='1';
    int counter=mymap->counter-'1';
    mymap->blank=' ';
    mymap->end='\n';
    mymap->zero=0;
    uChr      result1[BUFFERSIZE];
    chktoken (result1, TOKEN);
```

## myfork.c (2)

```
if (fork() == 0) {
    sleep(1);
    mymap->counter++;
    counter=mymap->counter-'1';
    chktoken (result1, TOKEN);
    if (fork() == 0) {
        sleep(1);
        mymap->counter++;
        counter=mymap->counter-'1';
        chktoken (result1, TOKEN);
        if (fork() == 0) {
            sleep(1);
            mymap->counter++;
            counter=mymap->counter-'1';
            chktoken (result1, TOKEN);
        }
        wait(NULL);
    }
    wait(NULL);
}
wait(NULL);
strcpy(mymap->stamp[counter], result1);
strcat(mymap->stamp[counter], " ");
printf("PID[%d] [%s]-[%d]\n", getpid(), result1, counter);
wait(NULL);
}
```

# mytest.c (1)

```
/*
 * Copyright (C) 2021-2021 Rahmat M. Samik-Ibrahim
 * http://rahmatm.samik-ibrahim.vlsm.org/
 * This program is free script/software. This program is distributed in the
 * hope that it will be useful, but WITHOUT ANY WARRANTY; without even the
 * implied warranty of MERCHANTABILITY or FITNESS FOR A PARTICULAR PURPOSE.
# INFO: TOP (Table of Processes)
 * REV01 Sun 04 Apr 2021 00:11:59 WIB
 * START Sat 03 Apr 2021 15:10:28 WIB
 */

#include "mylib.h"

int main(void) {
    uChr    result1[BUFFERSIZE];
    chktoken (result1, TOKEN);
    printf("%s\n", result1);
    uChr    result2[BUFFERSIZE];
    verifyToken (result2, TOKEN, result1);
    printf("%s: %s\n", TOKEN, result2);
    verifyToken (result2, "DODOLGRT", "rms46 0605-0687");
    printf("%s: %s\n", "DODOLGRT", result2);
    verifyToken (result2, "DODOLGRT", "rms46 1820-2A46");
    printf("%s: %s\n", "DODOLGRT", result2);
    sleep (1);
    chktoken (result1, TOKEN);
    printf("%s\n", result1);
    pickToken(result2, result1);
    printf("%s\n", result2);
}
```

# mytest.sh (1)

```
#!/bin/bash
# REVO1 Mon  5 Apr 17:08:58 WIB 2021
# START Sun  4 Apr 17:22:46 WIB 2021
# Copyright (C) 2021-2021 Rahmat M. Samik-Ibrahim http://rahmatm.samik-ibrahim.vlsm.org/
# This program is free script/software. This program is distributed in the
# hope that it will be useful, but WITHOUT ANY WARRANTY; without even the
# implied warranty of MERCHANTABILITY or FITNESS FOR A PARTICULAR PURPOSE.
# INFO: myfork00
```

```
CLEANFILE="WEEK06-MEMORY-SHARE.txt"
WEEKFILE="WEEK06-MEMORY-SHARE.bin"
```

```
TOKEN="OS212W06"
```

```
[ -f $CLEANFILE ] || { echo "No $CLEANFILE"; exit; }
```

```
sleep 1
echo "ZCZC $(date)"
echo -n "ZCZC $(./chktoken $TOKEN): "
echo "$(./verifyToken $TOKEN $(./chktoken $TOKEN))"
echo "ZCZC BINSIZE $(wc -c < $WEEKFILE)"
echo "ZCZC TXTSIZE $(wc -c < $CLEANFILE)"
FIRST=""
for II in $(cat $CLEANFILE) ; do
    [ ! -z "${II##*[!0-9]*}" ] && continue
    [ -z "$FIRST" ] && { FIRST=$II ; continue; }
    echo -n "ZCZC $FIRST $II: "
    echo "$(./verifyToken $TOKEN $FIRST $II)"
    FIRST=""
done
```



# Makefile (1)

```
# REV03 Mon 05 Apr 17:55:47 WIB 2021
# REV02 Sun 04 Apr 07:22:23 WIB 2021
# REV01 Sat 03 Apr 10:51:58 WIB 2021
# START Tue 13 Sep 11:44:18 WIB 2016
```

```
# INFO: With this "Makefile", just run:
# INFO:             make
```

```
CC           = gcc
CPP          = cpp
CFLAGS       = -std=gnu18
LDFLAGS      =
CPPFLAGS     =
DEPFLAGS     = -MM -MT $(@:.d=.o)
OUTPUT_OPTION = -o $@
COMPILE      = $(CC) $(DEPFLAGS) $(CFLAGS) $(CPPFLAGS) -c
SRCS         = $(wildcard *.c)
OBJ          = $(SRCS:.c=.o)
DEP          = $(OBJ:.o=.d)
PROGS        = $(SRCS:.c= )
```

```
P01=mytest
P02=chktoken
P03=verifyToken
P04=myfork
```

```
L99=mylib
WEEKFILE=WEEK06-MEMORY-SHARE.bin
CLEANFILE=WEEK06-MEMORY-SHARE.txt
```

# Makefile (2)

```
EXECS= \  
    $(P01) \  
    $(P02) \  
    $(P03) \  
    $(P04) \  
  
all: $(EXECS)  
  
test: $(EXECS)  
    ./$$(P04)  
    cat $(WEEKFILE) | wc -c > $(CLEANFILE)  
    cat $(WEEKFILE) | tr -dc '[:alnum:]\n -_' >> $(CLEANFILE)  
    bash mytest.sh  
  
$(EXECS): %: %.c $(DEPS) $(L99).c  
    $(CC) $(CFLAGS) $(L99).c $< -o $@ $(LDFLAGS)  
  
clean:  
    rm -f $(EXECS)  
    rm -f *.map  
    rm -f $(WEEKFILE) $(CLEANFILE)  
  
.phony: clean all test
```