## 06 PT - Processes & Threads Aneka Soal Ujian Sistem Operasi Rahmat M. Samik-Ibrahim et.al.

© 2016 - 2018 — Rev: 15 - 20-Mar-2018. Silakan mengubah, memperbanyak, serta mendistribusikan dokumen ini selama tidak menghapus ketentuan ini. URL: http://rms46.vlsm.org/2/201.pdf

## 1. **2016-1**

```
001 /* FORK
002 * (c) 2015-2016 M. Anwar Ma'sum and Rahmat M. Samik-Ibrahim
003 * This is a free software ----- Rev. 06 - 01-Apr-2016
004 */
005
006 #include <stdio.h>
007 #include <sys/types.h>
008 #include <unistd.h>
009
010 void main() {
011
      pid_t pid1, pid2, pid3;
012
      pid1 = pid2 = pid3 = getpid();
013
014
      printf(" 2016  2015  2014--\n========\n");
      printf("[%4d][%4d][%4d]\n", pid1, pid2, pid3);
015
016
      fork();
017
      pid1 = getpid();
018
      wait(NULL);
      pid2 = getpid();
019
020
      if(!fork()) {
021
         pid2 = getpid();
022
          fork();
023
024
      pid3 = getpid();
025
      wait(NULL);
026
      printf("[%4d][%4d]\n", pid1, pid2, pid3);
027 }
```

- (a) (KOLOM) Lingkari tahun angkatan anda berikut ini: (A) 2016 (B) 2015 (C) lainnya.
- (b) (BARIS) Lingkari sesuai angka terakhir (paling kanan) dari NPM anda: 0 1 2 3 4 5 6
- (c) Harap mengisi (KOLOM:BARIS) dengan 1000
- (d) Harap mengisi kolom dan baris lainnya sesuai dengan keluaran program di atas!

NPM		2016		2015		Lainnya	
0	[		] [		] [		]
1	[		] [		] [		]
2	[		] [		] [		]
3	[		] [		] [		]
4	[		] [		] [		]
5	[		] [		] [		]
6	[		] [		] [		]

## 2. **2016-2**

```
001 /*
002 * (c) 2016 Rahmat M. Samik-Ibrahim -- This is free software
003 * REV02 Tue Apr 11 19:09:21 WIB 2017
004 * START Sun Dec 04 00:00:00 WIB 2016
005 * wait()
                  = suspends until its child terminates.
                  = flushes the user-space buffers.
006 * fflush()
007 * getppid() = get parent PID
008 * ASSUME first pid > 2000; first ppid < 1000
009 */
010
011 #include <stdio.h>
012 #include <sys/types.h>
013 #include <unistd.h>
014 #include <sys/wait.h>
015 #define NN 2
016
017 void main (void) {
       int id1000=getpid()-1000;
       for (int ii=1; ii<=NN; ii++) {</pre>
019
020
          fork();
          wait(NULL);
021
022
          int rPID = getpid()-id1000; // "relative"
          int rPPID=getppid()-id1000; // "relative"
023
          if (rPPID < 1) rPPID=999;</pre>
024
          printf("Loop [%d] - rPID[%d] - rPPID[%4.4d]\n", ii, rPID, rPPID);
025
026
          fflush(NULL);
027
      }
028 }
```

Fill the following blanks (program output):

Loop [	] - rPID[	] - rPPID[	]
Loop [	] - rPID[	] - rPPID[	]
Loop [	] - rPID[	] - rPPID[	]
Loop [	] - rPID[	] - rPPID[	]
Loop [	] - rPID[	] - rPPID[	]
Loop [	] - rPID[	] - rPPID[	]

## 3. **2017-1**

```
Program Code of Processes and Threads
001 /*
                                                       019 #include <sys/wait.h>
002 * (c) 2005-2017 Rahmat M. Samik-Ibrahim
                                                       020 #include <stdlib.h>
003 * This is free software. Feel free to copy and/or
                                                       021
004 * modify and/or distribute it, provided this
                                                       022 void main(void) {
005 * notice, and the copyright notice, are preserved.
                                                       023
                                                              int firstPID = (int) getpid();
006 * REV02 Wed May 17 16:52:02 WIB 2017
                                                       024
                                                              int RelPID;
007 * REV00 Wed May 3 17:07:09 WIB 2017
                                                       025
* 800
                                                       026
                                                              fork();
009 * fflush(NULL): flushes all open output streams
                                                       027
                                                              wait(NULL);
                                                       028
                                                              fork();
010 * fork(): creates a new process by cloning
011 * getpid():
                 get PID (Process ID)
                                                       029
                                                              wait(NULL);
012 * wait(NULL): wait until the child is terminated
                                                       030
                                                              fork();
013 *
                                                       031
                                                              wait(NULL);
014 */
                                                       032
015
                                                       033
                                                              RelPID=(int)getpid()-firstPID+1000;
016 #include <stdio.h>
                                                              printf("RelPID: %d\n", RelPID);
                                                       034
017 #include <unistd.h>
                                                       035
                                                              fflush(NULL);
018 #include <sys/types.h>
                                                       036 }
```

Program Output (line 34 of every process):																	
R	е	1	Р	I	D	:					 						