

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(void) {
011     pid_t  pid1, pid2, pid3;
012
013     pid1 = pid2 = pid3 = getpid();
014     printf(" 2016  2015  Lainnya--\n=====\\n");
015     printf("[%4d] [%4d] [%4d]\\n", pid1, pid2, pid3);
016     fork();
017     pid1 = getpid();
018     wait(NULL);
019     pid2 = getpid();
020     if(!fork()) {
021         pid2 = getpid();
022         fork();
023     }
024     pid3 = getpid();
025     wait(NULL);
026     printf("[%4d] [%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	[]	[

[illegible]

3. 2017-1

Program Code of Processes and Threads	
<pre>001 /* 002 * (c) 2005-2017 Rahmat M. Samik-Ibrahim 003 * This is free software. Feel free to copy and/or 004 * modify and/or distribute it, provided this 005 * notice, and the copyright notice, are preserved. 006 * REV02 Wed May 17 16:52:02 WIB 2017 007 * REV00 Wed May 3 17:07:09 WIB 2017 008 * 009 * fflush(NULL): flushes all open output streams 010 * fork(): creates a new process by cloning 011 * getpid(): get PID (Process ID) 012 * wait(NULL): wait until the child is terminated 013 * 014 */ 015 016 #include <stdio.h> 017 #include <unistd.h> 018 #include <sys/types.h></pre>	<pre>019 #include <sys/wait.h> 020 #include <stdlib.h> 021 022 void main(void) { 023 int firstPID = (int) getpid(); 024 int RelPID; 025 026 fork(); 027 wait(NULL); 028 fork(); 029 wait(NULL); 030 fork(); 031 wait(NULL); 032 033 RelPID=(int)getpid()-firstPID+1000; 034 printf("RelPID: %d\n", RelPID); 035 fflush(NULL); 036 }</pre>
Program Output (line 34 of every process):	
<pre>R e l P I D : ----- -----</pre>	

4. 2017-2

The Program Code	
<pre>001 /* 002 * (c) 2017 Rahmat M. Samik-Ibrahim 003 * http://rahmatm.samik-ibrahim.vlsm.org/ 004 * This is free software. 005 * REV02 Mon Dec 11 17:46:01 WIB 2017 006 * START Sun Dec 3 18:00:08 WIB 2017 007 */ 008 009 #include <stdio.h> 010 #include <unistd.h> 011 #include <sys/types.h> 012 #include <sys/wait.h> 013 014 #define LOOP 3 015 #define OFFSET 1000</pre>	<pre>017 void main(void) { 018 int basePID = getpid() - OFFSET; 019 020 for (int ii=0; ii < LOOP; ii++) { 021 if(!fork()) { 022 printf("PID[%d]-PPID[%d]\n", 023 getpid() - basePID, 024 getppid() - basePID); 025 fflush(NULL); 026 } 027 wait(NULL); 028 }</pre>
Program Output (line 22 of every process):	
<pre>----- ----- -----</pre>	

5. 2018-1

```

01  /*
02  Copyright 2018 Rahmat M. Samik-Ibrahim
03  You are free to SHARE (copy and
04  redistribute the material in any medium
05  or format) and to ADAPT (remix,
06  transform, and build upon the material
07  for any purpose, even commercially).
08  This program is distributed in the hope
09  that it will be useful, but WITHOUT ANY
10  WARRANTY; without even the implied
11  warranty of MERCHANTABILITY or FITNESS
12  FOR A PARTICULAR PURPOSE.
13
14  * REV02 Wed May  2 11:30:19 WIB 2018
15  * START Wed Apr 18 19:50:01 WIB 2018
16  */
17
18  // DO NOT USE THE SAME SEMAPHORE NAME!!!!
19  // Replace "demo" with your own SSO name.
20  #define SEM_COUNT1      "/count-1-demo"
21  #define SEM_COUNT2      "/count-2-demo"
22  #define SEM_Mutex       "/mutex-demo"
23  #define SEM_SYNC        "/sync-demo"
24
25  #include <fcntl.h>
26  #include <stdio.h>
27  #include <stdlib.h>
28  #include <unistd.h>
29  #include <semaphore.h>
30  #include <sys/mman.h>
31  #include <sys/types.h>
32  #include <sys/wait.h>
33
34  // Shared Memory: R/W with no name.
35  #define PROT      (PROT_READ | PROT_WRITE)
36  #define VISIBLE   (MAP_ANONYMOUS|MAP_SHARED)
37
38  #define LOOP      2
39  #define BUFSIZE   1
40
41  sem_t*   ctr_prod;
42  sem_t*   ctr_cons;
43  sem_t*   mutex;
44  sem_t*   ssync;
45  int*     product;
46
47  // WARNING: NO ERROR CHECK! ///////////////
48  void flushprintf(char* str, int ii) {
49      printf("%s [%d]\n", str, ii);
50      fflush(NULL);
51  }
52
53  void init(void) {
54      product = mmap(NULL, sizeof(int),
55                      PROT, VISIBLE, 0, 0);
56      *product = 0;
57      ctr_prod = sem_open(SEM_COUNT1,
58                          O_CREAT, 0600, BUFSIZE);
59      ctr_cons = sem_open(SEM_COUNT2,
60                          O_CREAT, 0600, 0);
61      mutex     = sem_open(SEM_Mutex,
62                          O_CREAT, 0600, 1);
63      ssync     = sem_open(SEM_SYNC,
64                          O_CREAT, 0600, 0);
65  }
66
67  void producer (void) {
68      sem_wait(ssync);
69      flushprintf("PRODUCER  PID",getpid());
70      for (int loop=0; loop<LOOP; loop++) {
71          sem_wait(ctr_prod);
72          sem_wait(mutex);
73          flushprintf("PRODUCT  ",
74                      ++(*product));
75
76          sem_post(mutex);
77          sem_post(ctr_cons);
78      }
79      wait(NULL);
80  }
81
82  void consumer (void) {
83      flushprintf("CONSUMER  PID",getpid());
84      sem_post(ssync);
85      for (int loop=0; loop<LOOP; loop++) {
86          sem_wait(ctr_cons);
87          sem_wait(mutex);
88          flushprintf("CONSUME  ", *product);
89          sem_post(mutex);
90          sem_post(ctr_prod);
91      }
92  }
93
94  // WARNING: NO ERROR CHECK! ///////////////
95  void main(void) {
96      flushprintf("STARTING  PID",getpid());
97      init();
98      if (fork()) producer(); // Parent
99      else        consumer(); // Child
100      sem_unlink(SEM_COUNT1);
101      sem_unlink(SEM_COUNT2);
102      sem_unlink(SEM_SYNC);
103      sem_unlink(SEM_Mutex);
104      flushprintf("STOP HERE PID", getpid());
105  }

```

6. 2018-1 (continued)...

- (a) Assume the Parent PID is 1000 and the Child PID is 1001. What is the output of the program above?
- (b) Name all four (4) semaphore!
- (c) What is the purpose of line 68?
- (d) What is the purpose of line 71?

- (e) What is the purpose of line 77?
- (f) What is the purpose of line 84?
- (g) How many Critical Section(s) is/are there in the program above? Where/which lines are the Critical Section(s)?
- (h) Explain briefly the purpose of function `fflush(NULL)` in line 50!
- (i) What is the purpose of lines 98 - 101?

7. 2018-2

```

001 // FILE: 30-add1sub1.c =====
002 // Copyright (C) 2018 Rahmat M. Samik-Ibrahim.
003 /* You are free to SHARE (copy and redistribute the material in any medium or format) and to ADAPT (remix, transform, and build upon the material for any purpose, even commercially). 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. */

005 // REV04 Sun Dec 16 11:15:54 WIB 2018
006 // START Wed Nov 14 20:30:05 WIB 2018
008 #include <fcntl.h>
009 #include <stdio.h>
010 #include <stdlib.h>
011 #include <string.h>
012 #include <semaphore.h>
013 #include <unistd.h>
014 #include <sys/mman.h>
015 #include <sys/types.h>
016 #include <sys/stat.h>
017 #include <sys/wait.h>
019 #define MYFLAGS      O_CREAT | O_RDWR
020 #define MYPROTECT PROT_READ | PROT_WRITE
021 #define MYVISIBILITY      MAP_SHARED
022 #define SFILE          "demo-file.bin"
024 typedef struct {
025     sem_t  sync[3];
026     int    share;
027     int    loop;
028     pid_t  relative;
029 } myshare;
031 myshare* mymap;
033 void flushprintf(char* tag1, char* tag2){
034     printf("%s[%s] loop%d relative(%d)\n",
035           tag1, tag2, mymap->loop,
036           getpid() + mymap->relative);
037     fflush(NULL);
038 }

```

```

040 #define MAIN "30:ADDSUB"
041 #define ADD1 " 31:ADD1"
042 #define SUB1 " 32:SUB1"
044 void main(void) {
045     int fd =open(SFILE,MYFLAGS,S_IRWXU);
046     int ssize=sizeof(myshare);
047     truncate(SFILE, ssize);
048     mymap=mmap(NULL, ssize, MYPROTECT,
049               MYVISIBILITY, fd, 0);
050     mymap->share = 0;
051     mymap->loop = 3;
052     mymap->relative = 1000 - getpid();
053     sem_init (&(mymap->sync[0]), 1, 0);
054     sem_init (&(mymap->sync[1]), 1, 0);
055     sem_init (&(mymap->sync[2]), 1, 0);
056     flushprintf(MAIN, "EXEC");
057     if (!fork())
058         execlp("./31-add1", ADD1, NULL);
059     if (!fork())
060         execlp("./32-sub1", SUB1, NULL);
061     do {
062         sleep(1);
063         flushprintf(MAIN, "LOOP");
064     } while (--mymap->loop);
065     sem_wait (&(mymap->sync[0]));
066     sem_wait (&(mymap->sync[0]));
067     flushprintf(MAIN, "WAIT");
068     if (mymap->share > 1500)
069         flushprintf("SHARE +/-", "2000");
070     else if (mymap->share > 500)
071         flushprintf("SHARE +/-", "1000");
072     else
073         flushprintf("SHARE +/-", "0");
074     wait(NULL);
075     wait(NULL);
076     flushprintf(MAIN, "EXIT");
077     close(fd);
078 }

```

```

080 // FILE: 31-add1.c=====
081 // SEE ALSO: 30-add1sub1.c =====
082
083 void main(int argc, char* argv[]) {
084     int fd =open(SFILE,MYFLAGS,S_IRWXU);
085     int ssize=sizeof(myshare);
086     mymap=mmap(NULL, ssize, MYPROTECT,
087                MYVISIBILITY, fd, 0);
088     sem_post (&(mymap->sync[2]));
089     sem_wait (&(mymap->sync[1]));
090
091     sem_wait (&(mymap->sync[1]));
092     mymap->share=1000;
093     flushprintf(argv[0], "PASS");
094
095     while (mymap->loop) {
096         for(int ii=0; ii<1000000; ii++);
097         mymap->share++;
098     }
099     sem_post (&(mymap->sync[2]));
100     sem_wait (&(mymap->sync[1]));
101
102     flushprintf(argv[0], "EXIT");
103     sem_post (&(mymap->sync[2]));
104     sem_post (&(mymap->sync[0]));
105     close(fd);
106 }

```

```

105 // FILE: 32-sub1.c=====
106 // SEE ALSO: 30-add1sub1.c =====
107
108 void main(int argc, char* argv[]) {
109     int fd =open(SFILE,MYFLAGS,S_IRWXU);
110     int ssize=sizeof(myshare);
111     mymap=mmap(NULL, ssize, MYPROTECT,
112                MYVISIBILITY, fd, 0);
113     sem_post (&(mymap->sync[1]));
114     sem_wait (&(mymap->sync[2]));
115
116     mymap->share=2000;
117     flushprintf(argv[0], "PASS");
118     sem_post (&(mymap->sync[1]));
119     while (mymap->loop) {
120         for(int ii=0; ii<1000000; ii++);
121         mymap->share--;
122     }
123     sem_post (&(mymap->sync[1]));
124     sem_wait (&(mymap->sync[2]));
125     sem_wait (&(mymap->sync[2]));
126     flushprintf(argv[0], "EXIT");
127
128     sem_post (&(mymap->sync[0]));
129     close(fd);
130 }

```

- (a) What is the purpose of line 37?
- (b) Write the output of running program "**30-add1sub1**" (fill the blanks):

[illegible]

8. 2019-1 (83.3%)

```

001 // (c) 2019 This is Free Software R03
002 // Rahmat M. Samik-Ibrahim 20190508-0906
004 // WARNING: NO ERROR CHECK! ////////////////
005 // exit(STATUS) == exit with STATUS
006 // memcpy(*d,*s,n) == copy n from s to d
007 // mmap() == creates a new memory map
008 // usleep(DELAY1MS)== sleep 1 MS
010 #define TURNS 15
011 #define LAP 25
012 #define DELAY1MS 901
013 #define DELAY DELAY1MS*20
016 typedef struct {
017     char motoGP[35];
018     int countLap;
019 } drivers;
020 drivers D[]={
021     {"(93) M Marquez - Honda ", 0},
022     {"(42) A Rins - Suzuki ", 0},
023     {"(04) A Dovizioso - Ducati ", 0},
024     {"(46) V Rossi - Yamaha ", 0},
025     {"(09) D Petrucci - Ducati ", 0},
026     {"(12) M Vinales - Yamaha ", 0},
027     {"(43) J Miller - Ducati ", 0},
028     {"(30) T Nakagami - Honda ", 0},
029     {"(35) C Crutchlow - Honda ", 0},
030     {"(21) F Morbidelli - Yamaha ", 0},
031     {"(44) P Espargaro - KTM ", 0},
032     {"(41) A Espargaro - Aprilia", 0},
033     {"(21) F Quartararo - Yamaha ", 0},
034     {"(99) J Lorenzo - Honda ", 0},
035     {"(63) F Bagnaia - Ducati ", 0},
036     {"(36) J Mir - Suzuki ", 0},
037     {"(88) M Oliveira - KTM ", 0},
038     {"(05) J Zarco - KTM ", 0},
039     {"(06) S Bradl - Honda ", 0},
040     {"(29) A Iannone - Aprilia", 0},
041     {"(53) T Rabat - Ducati ", 0},
042     {"(17) K Abraham - Ducati ", 0},
043     {"(55) H Syahrin - KTM ", 0},
044     {"(38) B Smith - Aprilia", 0},
045 };
047 #include <semaphore.h>
048 #include <stdio.h>
049 #include <stdlib.h>
050 #include <string.h>
051 #include <sys/mman.h>
052 #include <sys/types.h>
053 #include <sys/wait.h>
054 #include <unistd.h>
055 #define SIZEofD (int) sizeof(D)
056 #define SIZEofD0 (int) sizeof(D[0])

```

```

057 #define NDRIVERS SIZEofD/SIZEofD0
058 typedef struct {
059     sem_t mutex;
060     sem_t turns[TURNS];
061     pid_t relPID;
062     volatile int rTime;
063     drivers D[NDRIVERS];
064 } shareMem;
065 #define MSIZE (int) sizeof(shareMem)
066 #define MAXSEM 2
067 #define MUTEX 1
068 #define PROTECT PROT_READ | PROT_WRITE
069 #define VISIBLE MAP_SHARED|MAP_ANONYMOUS
071 shareMem* mymap;
072 // =====
073 void init(void) {
074     printf("[1000] INIT: %d %d %d %d\n",
075         SIZEofD, SIZEofD0, NDRIVERS, MSIZE);
076     mymap=mmap(NULL, MSIZE, PROTECT,
077         VISIBLE, 0, 0);
078     for (int ii=0; ii<TURNS; ii++) {
079         sem_init (&(mymap->turns[ii]),
080             1, MAXSEM);
081     }
082     sem_init (&(mymap->mutex),1,MUTEX);
083     mymap->rTime=0;
084     mymap->relPID=getpid() - 1000;
085     memcpy(mymap->D, D, sizeof(D));
086     printf("[1000] INIT: END\n");
087 }
088 // =====
089 void motoGP(int number) {
090     pid_t relPID=getpid()-mymap->relPID;
091     while(mymap->D[number].countLap<LAP){
092         for (int ii=0; ii<TURNS; ii++) {
093             usleep(DELAY);
094             sem_wait (&(mymap->turns[ii]));
095             sem_post (&(mymap->turns[ii]));
096         }
097         mymap->rTime++;
098         mymap->D[number].countLap++;
099     }
100     sem_wait (&(mymap->mutex));
101     printf("[%d] %s Lap %2d rTime %3d\n",
102         relPID, mymap->D[number].motoGP,
103         mymap->D[number].countLap,
104         mymap->rTime++);
105     fflush(NULL);
106     sem_post (&(mymap->mutex));
107     exit (0);
108 } // (continued to the next page !!) //

```

```

110 void main(void) {
111     init();
112     printf("[1000] motoGP:START\n");
113     for (int ii=0; ii<NDRIVERS; ii++) {
114         if(!fork()) motoGP(ii);
115         usleep(DELAY1MS);
116     }
117     printf("[1000] motoGP:RACING\n");
118     for (int ii=0; ii<NDRIVERS; ii++)
119         wait(NULL);
120     printf("[1000] motoGP:FINISH\n");
121     exit (0);
122 }
////////////////////////////////////

```

- (a) **(Line 055)** (90%) SIZEofD = _____
- (b) **(Line 056)** (89%) SIZEofD0 = _____
- (c) **(Line 057)** (91%) NDRIVERS = _____
- (d) **(Line 065)** (88%) MSIZE = _____
- (e) What is the relative PID of the rider "**(41) A Espargaro**" (94%)? _____
- (f) Explain why the rider "**(93) M Marquez**" has the best chance to win! (86%)
- (g) Why is **(Line 120)** only executed by one process (i.e. **rPID 1000**) (54%)?

Program Output:

```

[1000] INIT: 960 40 24 1480
[1000] INIT: END
[1000] motoGP:START
[1000] motoGP:RACING
[1004] (46) V Rossi - Yamaha Lap 25 rTime 575
[1002] (42) A Rins - Suzuki Lap 25 rTime 577
[1001] (93) M Marquez - Honda Lap 25 rTime 579
[1003] (04) A Dovizioso - Ducati Lap 25 rTime 581
[1006] (12) M Vinales - Yamaha Lap 25 rTime 583
[1007] (43) J Miller - Ducati Lap 25 rTime 586
[1005] (09) D Petrucci - Ducati Lap 25 rTime 587
[1011] (44) P Espargaro - KTM Lap 25 rTime 589
[1009] (35) C Crutchlow - Honda Lap 25 rTime 593
[1010] (21) F Morbidelli - Yamaha Lap 25 rTime 594
[1008] (30) T Nakagami - Honda Lap 25 rTime 595
[1015] (63) F Bagnaia - Ducati Lap 25 rTime 597
[1013] (21) F Quartararo - Yamaha Lap 25 rTime 601
[1012] (41) A Espargaro - Aprilia Lap 25 rTime 602
[1014] (99) J Lorenzo - Honda Lap 25 rTime 603
[1018] (05) J Zarco - KTM Lap 25 rTime 605
[1017] (88) M Oliveira - KTM Lap 25 rTime 607
[1016] (36) J Mir - Suzuki Lap 25 rTime 609
[1019] (06) S Bradl - Honda Lap 25 rTime 612
[1020] (29) A Iannone - Aprilia Lap 25 rTime 613
[1023] (55) H Syahrin - KTM Lap 25 rTime 615
[1022] (17) K Abraham - Ducati Lap 25 rTime 617
[1021] (53) T Rabat - Ducati Lap 25 rTime 619
[1024] (38) B Smith - Aprilia Lap 25 rTime 621
[1000] motoGP:FINISH

```

9. 2019-2

```

001 // (c) 2019 This is Free Software R01
002 // Rahmat M. Samik-Ibrahim 20191209-1628
003 // R01 1215 1505
005 // File Name: 55-192a.c
006 // To run: ./55-192a
007
008 #include <fcntl.h>
009 #include <stdio.h>
010 #include <stdlib.h>
011 #include <string.h>
012 #include <semaphore.h>
013 #include <unistd.h>
014 #include <sys/mman.h>
015 #include <sys/types.h>
016 #include <sys/stat.h>
017 #include <sys/wait.h>

```

```

019 #define MYFLAGS O_CREAT | O_RDWR
020 #define MYPROTECT PROT_READ | PROT_WRITE
021 #define MYVISIBILITY MAP_SHARED
022 #define SFILE "demo-file.bin"
023
024 typedef struct {
025     sem_t sync1;
026     sem_t sync2;
027     pid_t relative;
028 } myshare;
029
030 myshare* mymap;
031
032 void flushprintf(char* tag){
033     printf("PIDr[%d] %s\n",
034         getpid() + mymap->relative, tag);
035     fflush(NULL);
036 }

```



```
038 void main(void) {
039     int fd  =open(SFILE,MYFLAGS,S_IRWXU);
040     int ssize=sizeof(myshare);
041     truncate(SFILE, ssize);
042     mmap=mmap(NULL, ssize, MYPROTECT,
043               MYVISIBILITY, fd, 0);
044     mmap->relative = 1000 - getpid();
045     sem_init (&(mmap->sync1), 1, 0);
046     sem_init (&(mmap->sync2), 1, 0);
047     flushprintf("START");
048     if (!fork())
049         execlp("./56-192b", "./56-192b", NULL);
050     wait(NULL);
051     flushprintf("EXIT");
052 }
053
054 // File Name: 56-192b.c
055 // To run: ./56-192b
056 // [line 1-37] see file 55-192a.c
```

```
059 void main(int argc, char* argv[]) {
060     int fd  =open(SFILE,MYFLAGS,S_IRWXU);
061     int ssize=sizeof(myshare);
062     mmap=mmap(NULL, ssize, MYPROTECT,
063               MYVISIBILITY, fd, 0);
064     flushprintf("START");
065     if(argc == 1) {
066         if (!fork()) {
067             sem_post (&(mmap->sync1));
068             sem_wait (&(mmap->sync2));
069             flushprintf("FORK CHILD");
070         } else {
071             sem_wait (&(mmap->sync1));
072             flushprintf("FORK PARENT");
073             sem_post (&(mmap->sync2));
074         }
075         execlp(argv[0], argv[0], "XYZZY", NULL);
076     }
077     wait(NULL);
078     flushprintf("EXIT");
079 }
```

OUTPUT of program "55-192a" (47.9%):