

PART II - CONCURRENT PROCESSES IN UNIX

Repository Link: <https://github.com/akshavib/SYSC4001-A2-P2.git>

1. Run two concurrent processes in C/C++ under Linux

Using the `fork` system call (page 472 of the Linux book), create two independent processes, which run indefinitely. Process 1 will run forever, will initialize a counter at 0, and will increment it in each cycle of an infinite loop. Process 2 will run forever, will initialize a counter at 0, and will increment it in each cycle of an infinite loop. Use delay functions to slow the display speed.

To finish the program, use the `kill` command (man pages), find the PID of both processes (`ps`) and kill them.

Two concurrent processes were successfully created and executed in C (*question1.c*) using the `fork()` system call under Linux. Process IDs (PIDs) for both the parent and child were identified using the `ps` command, and the processes were safely terminated using the `kill` command. The figures below show the execution and termination of both processes. The parent count begins very large because when the process forks, the child copies the parent's memory. This leads to both processes sharing the same print buffer which causes their outputs to overlap and mix together, making it look like the parent's count starts from a much larger number than it really does.

The `fork()` system call is used in Linux systems to create a new process. The new process is considered as the child process, while the one that invoked `fork()` is the parent process. When `fork()` is called by the parent process, the resulting child process receives its own Process ID, a copy of the parent's code, data, stack, and a separate memory space. The parent process will resume its execution, while the child process will start executing from the same point. The `fork()` function returns 0 to the child and provides the child's PID to the parent. This is the fundamental process that enables UNIX systems to launch new programs and support multitasking

```
Last login: Wed Nov  5 01:37:30 on ttys003
[akshavibaskaran@Akshavib-MBP ~ % cd desktop
[akshavibaskaran@Akshavib-MBP desktop % cd SYSC4001-A2-Part2
[akshavibaskaran@Akshavib-MBP SYSC4001-A2-Part2 % gcc question1.c -o question1
[akshavibaskaran@Akshavib-MBP SYSC4001-A2-Part2 % ./question1
Parent PID: 28000 , Count: #1
Child PID: 28001 , Count: #1
Child PID: 28001 , Count: #2
Parent PID: 28000 , Count: #2
Child PID: 28001 , Count: #3
Parent PID: 28000 , Count: #3
Child PID: 28001 , Count: #4
Parent PID: 28000 , Count: #4
Child PID: 28001 , Count: #5
Parent PID: 28000 , Count: #5
Child PID: 28001 , Count: #6
Parent PID: 28000 , Count: #6
Child PID: 28001 , Count: #7
Parent PID: 28000 , Count: #7
Child PID: 28001 , Count: #8
Parent PID: 28000 , Count: #8
Child PID: 28001 , Count: #9
Parent PID: 28000 , Count: #9
Child PID: 28001 , Count: #10
```

```

SYSC4001-A2-Part2 -- zsh -- 80x24
Child PID: 28001 , Count: #62
Parent PID: 28000 , Count: #62
Child PID: 28001 , Count: #63
Parent PID: 28000 , Count: #63
Child PID: 28001 , Count: #64
Parent PID: 28000 , Count: #64
Child PID: 28001 , Count: #65
Parent PID: 28000 , Count: #65
Child PID: 28001 , Count: #66
Parent PID: 28000 , Count: #66
Child PID: 28001 , Count: #67
Parent PID: 28000 , Count: #67
Child PID: 28001 , Count: #68
Parent PID: 28000 , Count: #68
Child PID: 28001 , Count: #69
Parent PID: 28000 , Count: #69
Child PID: 28001 , Count: #70
Parent PID: 28000 , Count: #70
Child PID: 28001 , Count: #71
Parent PID: 28000 , Count: #71
Child PID: 28001 , Count: #72
Parent PID: 28000 , Count: #72
zsh: terminated ./question1
akshavibaskaran@Akshavis-MBP SYSC4001-A2-Part2 % et

akshavibaskaran@Akshavis-MBP ~ % ps aux | grep question1
akshavibaskaran 28010 0.0 0.0 34128952 536 s003 R+ 10:17AM 0:00.01 grep question1
akshavibaskaran 28001 0.0 0.0 34130092 408 s002 S+ 10:16AM 0:00.00 ./question1
akshavibaskaran 28000 0.0 0.0 34128876 552 s002 S+ 10:16AM 0:00.01 ./question1
akshavibaskaran@Akshavis-MBP ~ % kill 28001 28000
akshavibaskaran@Akshavis-MBP ~ %

```

2. Extend the Processes. Now, process 1 will display only multiples of 3. That is, increment the counter in an infinite loop, calculate if it's a multiple of 3, and display that number. Also, display the number of cycles in each iteration. Example:

Cycle number: 0 – 0 is a multiple of 3

Cycle number: 1

Cycle number: 2

Cycle number: 3 – 3 is a multiple of 3

...

Process 2 will do the same but decrementing the value of the counter.

Use the exec system call to launch Process 2 (i.e., Process 2 should be a different program/executable).

Use delay functions to slower the display speed.

To finish the program execution, use the kill command (man pages), find the pid of both processes (ps) and kill them.

Both processes were successfully extended. The first program, *question2_p1.c*, implements an incrementing counter that displays multiples of three and uses the *exec* system call to launch the second program. The second program, *question2_p2.c*, executes as a separate process and prints multiples of three while decrementing its counter. As expected, when the parent process was terminated, the child process continued running independently, as shown in the second screenshot. The child process was then safely terminated using the appropriate commands.

The *exec()* system call function is used to replace and load a process' memory space with a new program and execute it without creating a new one. In general, it's called after *fork()* to overwrite the memory image that was copied over from the parent process to the child one.

```

Last login: Wed Nov  5 01:37:18 on ttys002
[akshavibaskaran@Akshavis-MBP ~ % cd desktop
[akshavibaskaran@Akshavis-MBP desktop % cd SYSC4001-A2-Part2
[akshavibaskaran@Akshavis-MBP SYSC4001-A2-Part2 % gcc question2_p1.c -o question2_p1
[akshavibaskaran@Akshavis-MBP SYSC4001-A2-Part2 % gcc question2_p2.c -o question2_p2
[akshavibaskaran@Akshavis-MBP SYSC4001-A2-Part2 % ./question2_p1
Cycle number: 0 - 0 is a multiple of 3
Cycle number: 0 - 0 is a multiple of 3
Cycle number: 1           12           if (pid < 0) { /* error occurred */
Cycle number: 1           12           printf(stderr, "Fork Failed\n");
Cycle number: 2           13           return 1;
Cycle number: 2           14       }
Cycle number: -2          15           if (pid ==0){
Cycle number: 3 - 3 is a multiple of 3
Cycle number: -3 - -3 is a multiple of 3 lp("./question2_p2", "question2_p2", NULL);
Cycle number: 4           17
Cycle number: -4          18       }
Cycle number: 5           18
Cycle number: -5          19       else{
Cycle number: 6 - 6 is a multiple of 3 t. count = 0;
Cycle number: -6 - -6 is a multiple of 3 e(1){
Cycle number: 7           22           if (count % 3 == 0){
Cycle number: 7           22           printf("Cycle number: %d - %d is a m
Cycle number: 8           23           }
Cycle number: -8          24       }

```

```

h2 ^~1 4 SYSC4001-A2-Part2 -- zsh -- 87x24
h2
h2 Cycle number: 33 - -33 is a multiple of 3 pid_t pid;
h2 Cycle number: -33 - -33 is a multiple of 3
h2 Cycle number: 34           8           /* fork the child process */
h2 Cycle number: -34          8           pid = fork();
h2 Cycle number: 35           9           Cycle number: 35
h2 Cycle number: -35          10          Cycle number: -35
h2 Cycle number: 36 - -36 is a multiple of 3 if (pid < 0) { /* error occurred */
h2 Cycle number: -36 - -36 is a multiple of 3   fprintf(stderr, "Fork Failed\n");
h2 Cycle number: 37           11           return 1;
h2 Cycle number: -37          12       }
h2 Cycle number: 38           13           if (pid ==0){
h2 Cycle number: 39 - 39 is a multiple of 3   execvp("./question2_p2", "question2_p2");
h2 zsh: terminated: ./question2_p1: killed by SIGKILL
h2 akshavibaskaran@Akshavis-MBP SYSC4001-A2-Part2 % Cycle number: -39 - -39 is a multiple
of 3
h2 Cycle number: -40          18           }
h2 Cycle number: -41          19           else{
h2 Cycle number: -42 - -42 is a multiple of 3   int count = 0;
h2 Cycle number: -43          21           while(1){
h2 Cycle number: -44          21           if (count % 3 == 0){
h2 Cycle number: -45 - -45 is a multiple of 3   printf("Cycle number: %d - %d is a m
h2 Cycle number: -46          22           }
h2 Cycle number: -47          23           }
h2

```

```

Last login: Wed Nov  5 01:37:23 on ttys002
[akshavibaskaran@Akshavis-MBP ~ % ps aux | grep question2_p1
akshavibaskaran 27021  0.0  0.0 34121080 568 s003 R+  1:38AM 0:00.01 g
rep question2_p1
akshavibaskaran 27018  0.0  0.0 34120876 552 s002 S+  1:38AM 0:00.01 .
/question2_p1
[akshavibaskaran@Akshavis-MBP ~ % kill 27018
[akshavibaskaran@Akshavis-MBP ~ % ps aux | grep question2_p2
akshavibaskaran 27024  0.0  0.0 34121136 712 s003 S+  1:38AM 0:00.01 g
rep question2_p2
akshavibaskaran 27019  0.0  0.0 34129068 560 s002 S  1:38AM 0:00.01 q
/question2_p2
akshavibaskaran 26947  0.0  0.0 34120876 552 ?? S  1:34AM 0:00.02 q
/question2_p2 ; /* using exec function to launch process 2 */
[akshavibaskaran@Akshavis-MBP ~ % kill 27019 26947
akshavibaskaran@Akshavis-MBP ~ %

```

```

h2 ^~1 4 SYSC4001-A2-Part2 -- zsh -- 87x24
h2
h2 Cycle number: -89          6           pid_t pid;
h2 Cycle number: -90 - -90 is a multiple of 3
h2 Cycle number: -91          8           /* fork the child process */
h2 Cycle number: -92          8           pid = fork();
h2 Cycle number: -93 - -93 is a multiple of 3 if (pid < 0) { /* error occurred */
h2 Cycle number: -94          9           fprintf(stderr, "Fork Failed\n");
h2 Cycle number: -95          10          return 1;
h2 Cycle number: -96 - -96 is a multiple of 3   }
h2 Cycle number: -97          11           if (pid ==0){
h2 Cycle number: -98          12           int count = 0;
h2 Cycle number: -99 - -99 is a multiple of 3   while(1){
h2 Cycle number: -100         13           if (count % 3 == 0){
h2 Cycle number: -101         15           else if (pid ==0){
h2 Cycle number: -102 - -102 is a multiple of 3   execvp("./question2_p2", "question2_p2");
h2 Cycle number: -103         17           }
h2 Cycle number: -104         18           }
h2 Cycle number: -105 - -105 is a multiple of 3   }
h2 Cycle number: -106         19           else{
h2 Cycle number: -107         20           int count = 0;
h2 Cycle number: -108 - -108 is a multiple of 3   while(1){
h2 Cycle number: -109         21           if (count % 3 == 0){
h2 Cycle number: -110         22           printf("Cycle number: %d - %d is a m
h2

```

```

Last login: Wed Nov  5 01:37:23 on ttys002
[akshavibaskaran@Akshavis-MBP ~ % ps aux | grep question2_p1
akshavibaskaran 27021  0.0  0.0 34121080 568 s003 R+  1:38AM 0:00.01 g
rep question2_p1
akshavibaskaran 27018  0.0  0.0 34120876 552 s002 S+  1:38AM 0:00.01 .
/question2_p1
[akshavibaskaran@Akshavis-MBP ~ % kill 27018
[akshavibaskaran@Akshavis-MBP ~ % ps aux | grep question2_p2
akshavibaskaran 27024  0.0  0.0 34121136 712 s003 S+  1:38AM 0:00.01 g
rep question2_p2
akshavibaskaran 27019  0.0  0.0 34129068 560 s002 S  1:38AM 0:00.01 q
/question2_p2
akshavibaskaran 26947  0.0  0.0 34120876 552 ?? S  1:34AM 0:00.02 q
/question2_p2 ; /* using exec function to launch process 2 */
[akshavibaskaran@Akshavis-MBP ~ % kill 27019 26947
akshavibaskaran@Akshavis-MBP ~ %

```

3. Extend the processes above once more. Use the wait system call. Process 1 starts as in 2, and when Process 2 starts, it waits for it. Process 2 runs until it reaches a value lower than -500. When this happens, Process 1 should end too.

The wait() system call was implemented in Process 1 to ensure it pauses execution until Process 2 completes its 500 iterations. Once Process 2 reached the specified termination condition (counter value below -500), Process 1 resumed and both processes were successfully terminated.

```

SYSC4001-A2-Part2 -- zsh -- 99x24
Last login: Wed Nov  5 10:42:02 on ttys002
[akshavibaskaran@Akshavis-MBP ~ % cd desktop
[akshavibaskaran@Akshavis-MBP SYSC4001-A2-Part2 % gcc question3_p1.c -o question3_p1
[akshavibaskaran@Akshavis-MBP SYSC4001-A2-Part2 % ./question3_p1
Cycle number: 0 - 0 is a multiple of 3
Cycle number: -1 = 0;
Cycle number: -2
Cycle number: -3 - -3 is a multiple of 3
Cycle number: -4 count % 3 == 0\{
Cycle number: -5 printf("Cycle number: %d - %d is a multiple of 3\n", count, count);
Cycle number: -6 - -6 is a multiple of 3
Cycle number: -7
Cycle number: -8
Cycle number: -9 - -9 is a multiple of 3 %d\n", count);
Cycle number: -10
Cycle number: -11
Cycle number: -12 - -12 is a multiple of 3
Cycle number: -13 10000); /* a delay for 0.01s */
Cycle number: -14
Cycle number: -15 - -15 is a multiple of 3
Cycle number: -16
Cycle number: -17
J

SYSC4001-A2-Part2 -- zsh -- 99x24
Cycle number: -480 - -480 is a multiple of 3
Cycle number: -481 >ypes.h>
Cycle number: -482 >n.h>
Cycle number: -483 - -483 is a multiple of 3
Cycle number: -484
Cycle number: -485
Cycle number: -486 - -486 is a multiple of 3
Cycle number: -487= 0;
Cycle number: -488
Cycle number: -489 - -489 is a multiple of 3
Cycle number: -490
Cycle number: -491 printf("Cycle number: %d - %d is a multiple of 3\n", count, count);
Cycle number: -492 - -492 is a multiple of 3
Cycle number: -493
Cycle number: -494
Cycle number: -495 - -495 is a multiple of 3
Cycle number: -496
Cycle number: -497
Cycle number: -498 - -498 is a multiple of 3
Cycle number: -499 10000); /* a delay for 0.01s */
Cycle number: -500
Child Process Complete.
Child Complete, Parent Process ending now.
akshavibaskaran@Akshavis-MBP SYSC4001-A2-Part2 %

```

4. Extend the processes above once more. They should now share memory. The primary functions are listed in the book and course materials. Using `shmget`, `shmat`, `shmdt`, and `shmctl`, add two common variables shared between the two processes. The first variable contains the value of the multiple (in the example above: 3; that number can be changed and then the program should adapt and display the multiples of the chosen number). The second variable contains the value of the counter used by Process 1, which is now shared with Process 2. Process 2 starts only when the value of this variable is larger than 100. Each of the processes should now react to the value of the shared variable and display a message identifying themselves and numbers in shared memory. Both processes finish when the value of the shared variable is larger than 500.

All required functionality has been successfully implemented. The two processes now share memory through the use of `shmget`, `shmat`, `shmdt`, and `shmctl`. Process 1 updates the counter stored in shared memory, while Process 2 monitors this value and begins execution only after the counter exceeds 100.

```

Last login: Thu Nov  6 15:19:27 on ttys000
akshavibaskaran@dhcp-42-43 ~ % cd desktop
akshavibaskaran@dhcp-42-43 desktop % cd SYSC4001-A2-P2
akshavibaskaran@dhcp-42-43 SYSC4001-A2-P2 % gcc question4_p1.c -o question4_p1
akshavibaskaran@dhcp-42-43 SYSC4001-A2-P2 % ./question4_p1
_Process1 || Cycle number: 0 is a multiple of 3
_Process1 || Cycle number: 2
_Process1 || Cycle number: 3 is a multiple of 3
_Process1 || Cycle number: 5
_Process1 || Cycle number: 6 is a multiple of 3
_Process1 || Cycle number: 8
_Process1 || Cycle number: 9 is a multiple of 3
_Process1 || Cycle number: 11
_Process1 || Cycle number: 12 is a multiple of 3
_Process1 || Cycle number: 14
_Process1 || Cycle number: 15 is a multiple of 3
_Process1 || Cycle number: 17
_Process1 || Cycle number: 18 is a multiple of 3
_Process1 || Cycle number: 20
_Process1 || Cycle number: 21 is a multiple of 3
_Process1 || Cycle number: 23
_Process1 || Cycle number: 24 is a multiple of 3
_Process1 || Cycle number: 26
_Process1 || Cycle number: 27 is a multiple of 3
_Process1 || Cycle number: 29
_Process1 || Cycle number: 30 is a multiple of 3

```

```

_Process1 || Cycle number: 89
_Process1 || Cycle number: 90 is a multiple of 3
_Process1 || Cycle number: 92
_Process1 || Cycle number: 93 is a multiple of 3
_Process1 || Cycle number: 95
_Process1 || Cycle number: 96 is a multiple of 3
_Process1 || Cycle number: 98
_Process1 || Cycle number: 99 is a multiple of 3
_Process2 || Cycle number: 101
_Process2 || Cycle number: 102 is a multiple of 3
_Process2 || Cycle number: 103 is a multiple of 3
_Process2 || Cycle number: 104
_Process2 || Cycle number: 105 is a multiple of 3
_Process2 || Cycle number: 106 is a multiple of 3
_Process2 || Cycle number: 107
_Process2 || Cycle number: 108 is a multiple of 3
_Process2 || Cycle number: 109
_Process2 || Cycle number: 110 is a multiple of 3
_Process2 || Cycle number: 111 is a multiple of 3
_Process2 || Cycle number: 112 is a multiple of 3
_Process2 || Cycle number: 113
_Process2 || Cycle number: 114 is a multiple of 3

```

```

_Process2_ || Cycle number: 483 is a multiple of 3
_Process1_ || Cycle number: 483 is a multiple of 3
_Process2_ || Cycle number: 485
_Process1_ || Cycle number: 485
_Process2_ || Cycle number: 486 is a multiple of 3
_Process1_ || Cycle number: 486 is a multiple of 3
_Process2_ || Cycle number: 488
_Process1_ || Cycle number: 488
_Process2_ || Cycle number: 489 is a multiple of 3
_Process1_ || Cycle number: 489 is a multiple of 3
_Process2_ || Cycle number: 491
_Process1_ || Cycle number: 491
_Process2_ || Cycle number: 492 is a multiple of 3
_Process1_ || Cycle number: 492 is a multiple of 3
_Process2_ || Cycle number: 494
_Process1_ || Cycle number: 494
_Process2_ || Cycle number: 495 is a multiple of 3
_Process1_ || Cycle number: 495 is a multiple of 3
_Process2_ || Cycle number: 497
_Process1_ || Cycle number: 497
_Process2_ || Cycle number: 498 is a multiple of 3
_Process1_ || Cycle number: 498 is a multiple of 3
_Process2_ || Cycle number: 500
_Process1_ || Cycle number: 500
Shared counter > 500. Process 2 finished.
Shared counter > 500. Process 1 and Process 2 finished.
akshavibaskaran@dhcp-42-43 SYSC4001-A2-P2 %

```

5. Extend the processes above once more. They should now protect concurrent access to the shared memory positions. On top of the shm instructions, you should protect the shared memory access using semaphores. Use semget, semop, semctl to protect the shared memory section.

The implementation has been successfully extended to include protection of concurrent access to the shared memory. The semaphores ensure that only one process accesses or modifies the shared variables at a time, preventing race conditions and ensuring data consistency. The program functions correctly, with both processes synchronizing properly through the use of semaphores.

```

Last login: Thu Nov  6 15:28:26 on ttys000
akshavibaskaran@dhcp-42-43 ~ % cd desktop
akshavibaskaran@dhcp-42-43 desktop % cd SYSC4001-A2-P2
akshavibaskaran@dhcp-42-43 SYSC4001-A2-P2 % gcc question5_p1.c -o question5_p1
akshavibaskaran@dhcp-42-43 SYSC4001-A2-P2 % gcc question5_p2.c -o question5_p2
akshavibaskaran@dhcp-42-43 SYSC4001-A2-P2 % ./question5_p1
_Process1_ || Cycle number: 0 is a multiple of 3
_Process1_ || Cycle number: 2
_Process1_ || Cycle number: 3 is a multiple of 3
_Process1_ || Cycle number: 5
_Process1_ || Cycle number: 6 is a multiple of 3
_Process1_ || Cycle number: 8
_Process1_ || Cycle number: 9 is a multiple of 3
_Process1_ || Cycle number: 11
_Process1_ || Cycle number: 12 is a multiple of 3
_Process1_ || Cycle number: 14
_Process1_ || Cycle number: 15 is a multiple of 3
_Process1_ || Cycle number: 17
_Process1_ || Cycle number: 18 is a multiple of 3
_Process1_ || Cycle number: 20
_Process1_ || Cycle number: 21 is a multiple of 3
_Process1_ || Cycle number: 23
_Process1_ || Cycle number: 24 is a multiple of 3
_Process1_ || Cycle number: 26
_Process1_ || Cycle number: 27 is a multiple of 3
_Process1_ || Cycle number: 29
_Process1_ || Cycle number: 30 is a multiple of 3
_Process1_ || Cycle number: 32
_Process1_ || Cycle number: 33 is a multiple of 3
_Process1_ || Cycle number: 35
_Process1_ || Cycle number: 36 is a multiple of 3
_Process1_ || Cycle number: 38
_Process1_ || Cycle number: 39 is a multiple of 3
_Process1_ || Cycle number: 41
_Process1_ || Cycle number: 42 is a multiple of 3

```

```

_Process2_ || Cycle number: 477 is a multiple of 3
_Process2_ || Cycle number: 477 is a multiple of 3
_Process2_ || Cycle number: 479
_Process1_ || Cycle number: 479
_Process2_ || Cycle number: 480 is a multiple of 3
_Process1_ || Cycle number: 480 is a multiple of 3
_Process2_ || Cycle number: 482
_Process1_ || Cycle number: 482
_Process2_ || Cycle number: 483 is a multiple of 3
_Process1_ || Cycle number: 483 is a multiple of 3
_Process2_ || Cycle number: 485
_Process1_ || Cycle number: 485
_Process2_ || Cycle number: 486 is a multiple of 3
_Process1_ || Cycle number: 486 is a multiple of 3
_Process2_ || Cycle number: 488
_Process1_ || Cycle number: 488
_Process2_ || Cycle number: 489 is a multiple of 3
_Process1_ || Cycle number: 489 is a multiple of 3
_Process2_ || Cycle number: 491
_Process1_ || Cycle number: 491
_Process2_ || Cycle number: 492 is a multiple of 3
_Process1_ || Cycle number: 492 is a multiple of 3
_Process2_ || Cycle number: 494
_Process1_ || Cycle number: 494
_Process2_ || Cycle number: 495 is a multiple of 3
_Process1_ || Cycle number: 495 is a multiple of 3
_Process2_ || Cycle number: 497
_Process1_ || Cycle number: 497
_Process2_ || Cycle number: 498 is a multiple of 3
_Process1_ || Cycle number: 498 is a multiple of 3
_Process2_ || Cycle number: 500
_Process1_ || Cycle number: 500
Shared counter > 500. Process 2 finished.
Shared counter > 500. Process 1 and Process 2 finished.
akshavibaskaran@dhcp-42-43 SYSC4001-A2-P2 %

```