1η Εργαστηριακή Άσκηση

«Λειτουργικά Συστήματα» (Ε' ΕΞΑΜΗΝΟ)

Ον/μο: **ΠΡΟΚΟΠΙΟΣ ΚΑΜΑΤΣΟΣ**

ΑΜ: **1072586**

Ον/μο: **ΧΡΥΣΟΣΤΟΜΟΣ-ΑΘΑΝΑΣΙΟΣ ΚΑΤΣΙΓΙΑΝΝΗΣ**

ΑΜ: **1072490**

Ον/μο: **ΚΡΙΣΤΙΑΝ ΛΟΥΚΑ**

ΑΜ: **1072625**

Ον/μο: **ΑΡΓΥΡΗΣ ΣΟΦΟΤΑΣΙΟΣ**

ΑΜ: **1079616**

Μέρος 1

Ερώτημα Α

Το πρώτο πρόγραμμα ξεκινά ορίζοντας την ακέραια μεταβλητή i, η οποία χρησιμοποιείται ως μετρητής στο for loop που ακολουθεί και μια επίσης ακέραια μεταβλητή με το όνομα pid η οποία παρακάτω γίνεται αποδέκτης της τιμής που επιστρέφει η fork()αποθηκεύοντας το id της διεργασίας που δημιουργείται.

Η κλήση συστήματος fork() δημιουργεί ένα ακριβές αντίγραφο της αρχικής διεργασίας (μητρική/γονική - parent process), που ονομάζεται θυγατρική διεργασία (child process) και επιστρέφει την τιμή 0 στη θυγατρική και το pid της θυγατρικής στη μητρική διεργασία.

Από εκείνο το σημείο κι έπειτα οι δύο διεργασίες εκτελούνται παράλληλα.

Η γονική διεργασία κοιμάται για 2 secs και μετά τερματίζει την εκτέλεσή της. Η θυγατρική μπαίνει σε ένα for loop που εκτελεί 3 επαναλήψεις: σε κάθε επανάληψη τυπώνει μέσω της getppid() το pid της μητρικής διεργασίας και διακόπτεται (κοιμάται) για 1 sec. Στις πρώτες δύο επαναλήψεις όλα εξελίσσονται όπως αναμενόταν, ωστόσο στην τρίτη επανάληψη λόγω του ότι έχουμε διανύσει τα δύο πρώτα secs και 'βαδίζουμε' στο τρίτο, η γονική διεργασία έχει τερματιστεί κι έτσι η συνάρτηση getppid() δεν μπορεί να επιστρέψει το pid της γονικής. Η τιμή που επιστρέφει η getppid() είναι 1 και ο λόγος είναι ο εξής:

Οι διεργασίες στο UNIX είναι οργανωμένες σε ένα δένδρο διεργασιών. Στη ρίζα του δένδρου αυτού βρίσκεται η διεργασία με όνομα init, η οποία είναι ουσιαστικά η αρχική διεργασία του λειτουργικού συστήματος. Αυτό οδηγεί σε μια ιεραρχική οργάνωση του συνόλου των διεργασιών που εκτελούνται. Στην περίπτωση όπου η γονική διεργασία τερματίσει την εκτέλεσή της πριν ολοκληρωθεί η θυγατρική τότε τη θυγατρική διεργασία κληρονομεί η init.

Έτσι, στο πρόγραμμά μας η θυγατρική διεργασία υφίσταται μια αλλαγή γονικής διεργασίας στη διεργασία init που έχει pid = 1 και είναι η τιμή που επιστρέφει η getppid() στην τρίτη επανάληψη του for loop.

Στη συνέχεια απεικονίζεται η έξοδος του προγράμματος:

```
My parent is 25272
My parent is 25272
My parent is 1
```

Το δεύτερο πρόγραμμα ξεκινά όπως και το προηγούμενο ορίζοντας την ακέραια μεταβλητή i η οποία χρησιμοποιείται ως μετρητής στο for loop που ακολουθεί και την επίσης ακέραια μεταβλητή pid η οποία παρακάτω αποθηκεύει την τιμή που επιστρέφει η fork().

Από εκείνο το σημείο κι έπειτα οι δύο διεργασίες εκτελούνται παράλληλα, εισέρχονται στο for loop και εκτελούν 500 επαναλήψεις. Σε κάθε επανάληψη τυπώνουν τον αριθμό της επανάληψης ξεκινώντας από το 1 και φτάνοντας έως και το 500. Επειδή έχουμε δύο διεργασίες, τυπώνονται από δύο φορές οι αριθμοί 1, 2, …., 500. Με κάθε εκτύπωση τυπώνεται ο αριθμός και η ιδιότητα (parent ή child) της διεργασίας που εκτελείται.

Η σειρά με την οποία τυπώνονται οι αριθμοί, αν δηλαδή προέρχονται από τη μητρική ή τη θυγατρική διεργασία είναι τυχαία

και αλλάζει κάθε φορά που τρέχουμε το πρόγραμμα, καθώς οι δύο διεργασίες 'μοιράζονται' το for loop (πχ. η μητρική τυπώνει έναν ή περισσότερους αριθμούς στη σειρά, ακολουθεί η θυγατρική κ.ο.κ.).

Στη συνέχεια απεικονίζονται λόγω όγκου μόνο οι πρώτες και οι τελευταίες γραμμές από την έξοδο του προγράμματος:

```
  1 (parent)
  2 (parent)
  3 (parent)
  4 (parent)
  5 (parent)
  6 (parent)
  7 (parent)
  8 (parent)
  9 (parent)
 10 (parent)
 11 (parent)
 12 (parent)
 13 (parent)
 14 (parent)
 15 (parent)
 16 (parent)
 17 (parent)
 18 (parent)
 19 (parent)
 20 (parent)
 21 (parent)
  1 (child)
 22 (parent)
 23 (parent)
 24 (parent)
  2 (child)
 25 (parent)
  3 (child)
 26 (parent)
  4 (child)
 27 (parent)
```

...

```
498 (parent)
473 (child)
499 (parent)
474 (child)
500 (parent)
475 (child)
476 (child)
477 (child)
478 (child)
479 (child)
480 (child)
481 (child)
482 (child)
483 (child)
484 (child)
485 (child)
486 (child)
487 (child)
488 (child)
489 (child)
490 (child)
491 (child)
492 (child)
493 (child)
494 (child)
495 (child)
496 (child)
497 (child)
498 (child)
499 (child)
500 (child)
```

Ερώτημα Β

Για όλα τα δυνατά σενάρια σειριακής και παράλληλης εκτέλεσης, η ελάχιστη τιμή της κοινά διαμοιραζόμενης μεταβλητής X είναι 500, ενώ η μέγιστη τιμή της είναι 1.000.

Η ελάχιστη τιμή 500 προκύπτει στο παρακάτω σενάριο:

Αρχικά X = 0.

Η θυγατρική διεργασία P_1 ξεκινάει την εκτέλεσή της και εκτελεί τις πρώτες δύο εντολές του for loop. Έτσι, η τοπική στην P_1 μεταβλητή tmp = 1.

Στη σημείο αυτό η διεργασία P_1 χάνει τη CPU (πχ. εκτελεί μια εντολή sleep()) και την παίρνει η θυγατρική διεργασία P_2 η οποία εκτελεί επίσης τις πρώτες δύο εντολές του for loop. Έτσι, η τοπική στην P_2 μεταβλητή tmp = 1.

Η διεργασία P_1 ξαναπαίρνει τη CPU και εκτελεί την $3^η$ εντολή του for loop θέτοντας X = 1.

Στη συνέχεια ξαναπαίρνει τη CPU η διεργασία P_2, εκτελεί την $3^η$ εντολή του for loop θέτοντας εκ νέου τη X = 1.

Άρα, στο παραπάνω σενάριο μετά την πρώτη επανάληψη του for loop και στις δύο διεργασίες X = 1.

Αυτό επαναλαμβάνεται για 500 επαναλήψεις δίνοντας τελικά ελάχιστη τιμή X = 500.

Η μέγιστη τιμή 1.000 προκύπτει στο σενάριο κατά το οποίο η διεργασία που κατέχει τη CPU εκτελεί ακέραιο αριθμό επαναλήψεων του for loop (δηλ. εκτελεί και τις τρεις εντολές του for loop πριν χάσει τη CPU).

Αν πχ. ξεκινάει πρώτη η P_1 και εκτελεί k_1 επαναλήψεις, τότε τη στιγμή που χάνει τη CPU X = k_1, στη συνέχεια η P_2 παίρνει τη CPU και εκτελεί m_1 επαναλήψεις οπότε X = k_1 + m_1, στη συνέχεια παίρνει τη CPU πάλι η P_1 για να εκτελέσει τις επόμενες k_2 επαναλήψεις οπότε X = k_1 + m_1 + k_2 κ.ο.κ. Μετά και την $500^η$ επανάληψη κάθε διεργασίας θα ισχύει X = 500 + 500 = 1.000.

Πρόγραμμα

Χρησιμοποιούμε τρεις διαμοιραζόμενες μεταβλητές: τη X και τις μεταβλητές turn και interested[2] του αλγόριθμου του Peterson (int *pX, *pturn, *pinterested στον κώδικα) [θα μπορούσαμε να χρησιμοποιήσουμε και μια διαμοιραζόμενη δομή τύπου struct με πεδία τις τρεις μεταβλητές]. Οι εντολές enter_region() και leave_region() τοποθετούνται πριν την πρώτη και μετά την τελευταία εντολή του σώματος εντολών του for loop.

Ο κώδικας είναι ο εξής:

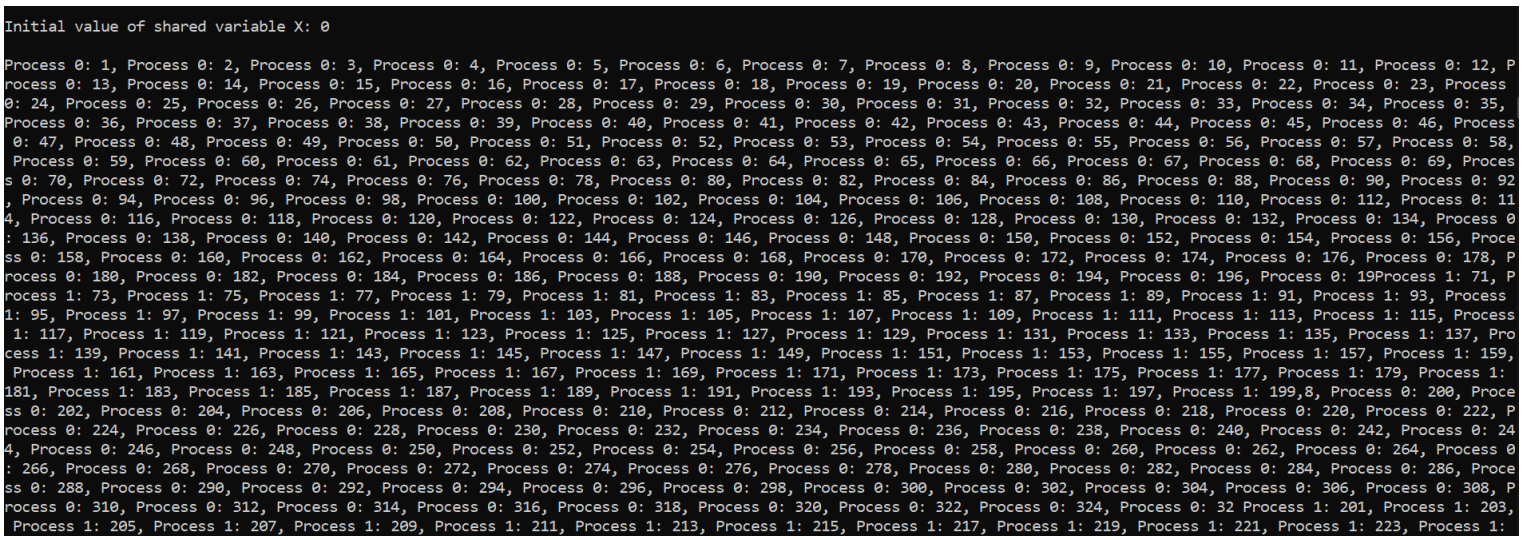
```
#include <stdio.h>        /* printf()                */
#include <stdlib.h>       /* exit()                  */
#include <sys/types.h>    /* key_t, sem_t, pid_t     */
#include <sys/shm.h>      /* shmat(), IPC_RMID       */
#include <sys/wait.h>     /* waitpid()               */
#include <errno.h>        /* errno, ECHILD           */
#include <fcntl.h>        /* O_CREAT, O_EXEC         */

void child_process_i ();
void enter_region ();
void leave_region ();

int *pX;  /* shared var */
int *pturn;  /* Peterson's algorithm shared vars */
int *pinterested;


int main ()
{
      pid_t w;
      int status;

      pid_t pid[10];    /* fork pids */
      int i;  /* counter var */


      /* initialize shared variables in shared memory */

      int shmid1, shmid2, shmid3;  /* shared memory ids */

      shmid1 = shmget (IPC_PRIVATE, sizeof (int), 0644 | IPC_CREAT);
      shmid2 = shmget (IPC_PRIVATE, sizeof (int), 0644 | IPC_CREAT);
      shmid3 = shmget (IPC_PRIVATE, sizeof (int) * 2, 0644 | IPC_CREAT);

      /* shared memory error check */
       if (shmid1 < 0 || shmid2 < 0 || shmid3 < 0)
      {
```

```
      perror ("\nShmget error.\n");
      exit (1);
 }

/* attach pX, pturn, pinterested to shared memory */
pX = (int *) shmat (shmid1, NULL, 0);
pturn = (int *) shmat (shmid2, NULL, 0);
pinterested = (int *) shmat (shmid3, NULL, 0);

/* initialize shared variables */
*pX = 0;
for (i = 0; i < 2; i++)
      *(pinterested + i) = 0;

printf("\n\nInitial value of shared variable X: %d\n\n", *pX);


for (i = 0; i < 2; i++)
{
      pid[i] = fork ();  /* creation of child processes */

      if (pid[i] < 0)
      {
            printf ("\nFork error.\n");
            return (-1);
      }
      else if (pid[i] == 0)  /* child process */
            {
                  child_process_i (i);
                  exit (0);
            }
}


/* parent process */
for (i = 0; i < 2; i++)
{
```

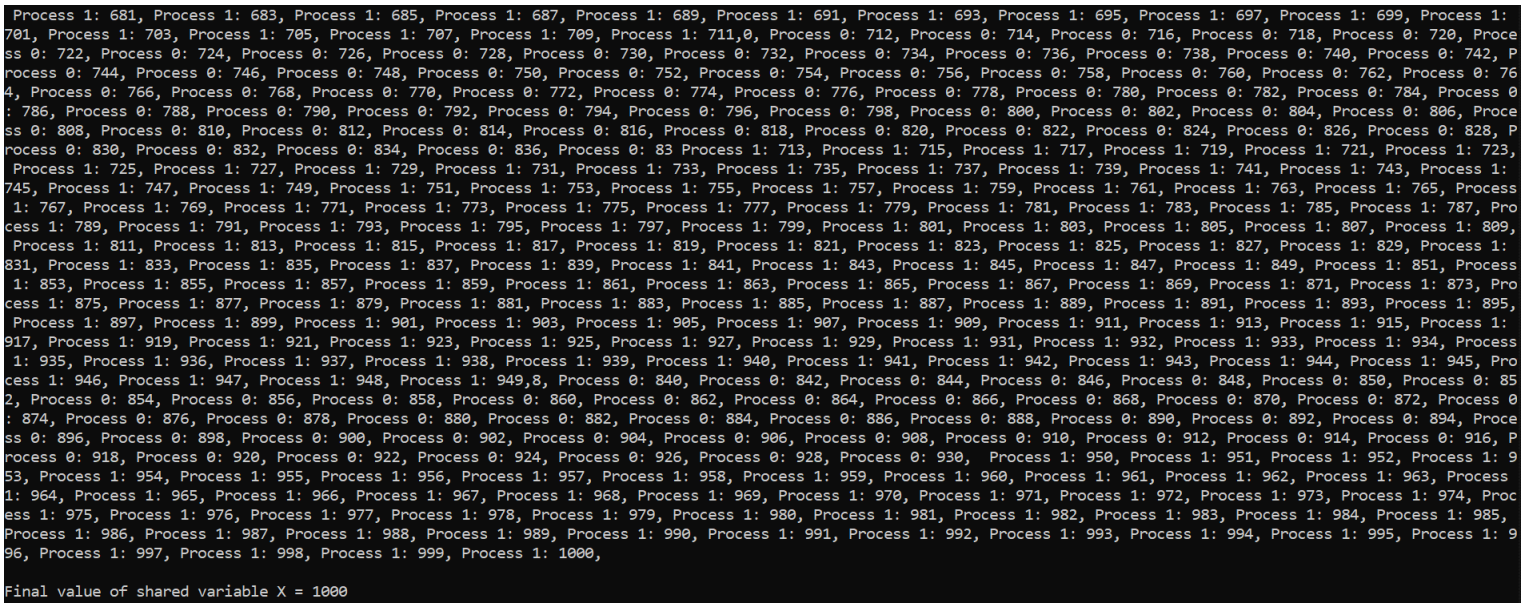
```
            /* wait both child processes to exit */
            w = waitpid (pid[i], &status, WUNTRACED | WCONTINUED);
            if (w == -1)
            {
                  perror ("\nWaitpid error.\n");
                  status = -1;
                  return status;
            }
            // if (WIFEXITED(status))  /* check if the two child processes
               end normally */
                  //  printf("\n\nChild  %d  terminated  with  exit  status
                     %d.\n", w, WEXITSTATUS(status));
            // else
                  // printf("\n\nChild %d terminated abnormally.\n", w);
      }


      printf ("\n\nFinal value of shared variable X = %d\n\n\n", *pX);

      /* shared memory detach */
      shmdt (pX);
      shmctl (shmid1, IPC_RMID, 0);
      shmdt (pturn);
      shmctl (shmid2, IPC_RMID, 0);
      shmdt (pinterested);
      shmctl (shmid3, IPC_RMID, 0);

      exit (0);
}


/* increment by 1 shared variable X for 500 times */
void child_process_i (int p)
{
      int tmp, i;  /* local vars */

      for (i = 1; i <= 500; i++)
      {
```

```
            enter_region (p);
            tmp = *pX;
            tmp = tmp + 1;
            *pX = tmp;
            printf("Process %d: %d, ", p, *pX);
            leave_region (p);
      }
}


/* enter_region() of Peterson's algorithm */
void enter_region (int process)
{
      int other = 1 - process;
      *(pinterested + process) = 1;
      *pturn = process;

      while ((*pturn == process) && (*(pinterested + other) == 1))
            ;
}


/* leave_region() of Peterson's algorithm */
void leave_region (int process)
{
      *(pinterested + process) = 0;
}
```

Το αντίστοιχο αρχείο του προγράμματος είναι το **1.B.c** και στη συνέχεια απεικονίζονται οι πρώτες και οι τελευταίες γραμμές (αυτό λόγω όγκου) από την έξοδο του προγράμματος:



...



Όπως φαίνεται στο screenshot, οι δύο διεργασίες από ένα σημείο κι έπειτα και για ένα μεγάλο αριθμό επαναλήψεων του for loop εκτελούν μόνο μία επανάληψη κάθε φορά που παίρνουν τη CPU.

Για να δούμε πώς επιτυγχάνεται η ελάχιστη τιμή 500, απενεργοποιούμε τις κλήσεις στις εντολές enter_region() και leave_region() θέτοντάς τες σε σχόλια και τοποθετούμε μετά την εντολή tmp = tmp + 1; και πριν την εντολή *pX = tmp; την εντολή

usleep(100). Τότε η έξοδος του προγράμματος (πρώτες και τελευταίες γραμμές) είναι η εξής:

```
Initial value of shared variable X: 0

Process 0: 1, Process 0: 2, Process 0: 3, Process 0: 4, Process 0: 5, Process 0: 6, Process 0: 7, Process 0: 8, Process 0: 9, Process 0: 10, Process 0: 11, Process 0: 12, P
rocess 0: 13, Process 0: 14, Process 0: 15, Process 0: 16, Process 0: 17, Process 0: 18, Process 0: 19, Process 0: 20, Process 0: 21, Process 0: 22, Process 0: 23, Process
0: 24, Process 0: 25, Process 0: 26, Process 0: 27, Process 0: 28, Process 0: 29, Process 0: 30, Process 0: 31, Process 0: 32, Process 0: 33, Process 0: 34, Process 0: 35,
Process 0: 36, Process 0: 37, Process 0: 38, Process 0: 39, Process 0: 40, Process 0: 41, Process 0: 42, Process 0: 43, Process 0: 44, Process 0: 45, Process 0: 46, Process
 0: 47, Process 0: 48, Process 0: 49, Process 0: 50, Process 0: 51, Process 0: 52, Process 0: 53, Process 0: 54, Process 0: 55, Process 0: 56, Process 0: 57, Process 0: 58,
 Process 0: 59, Process 0: 60, Process 0: 61, Process 0: 62, Process 0: 63, Process 0: 64, Process 0: 65, Process 0: 66, Process 0: 67, Process 0: 68, Process 0: 69Process
1: 1, Process 1: 2, Process 1: 3, Process 1: 4, Process 1: 5, Process 1: 6, Process 1: 7, Process 1: 8, Process 1: 9, Process 1: 10, Process 1: 11, Process 1: 12, Process 1
: 13, Process 1: 14, Process 1: 15, Process 1: 16, Process 1: 17, Process 1: 18, Process 1: 19, Process 1: 20, Process 1: 21, Process 1: 22, Process 1: 23, Process 1: 24, P
rocess 1: 25, Process 1: 26, Process 1: 27, Process 1: 28, Process 1: 29, Process 1: 30, Process 1: 31, Process 1: 32, Process 1: 33, Process 1: 34, Process 1: 35, Process
1: 36, Process 1: 37, Process 1: 38, Process 1: 39, Process 1: 40, Process 1: 41, Process 1: 42, Process 1: 43, Process 1: 44, Process 1: 45, Process 1: 46, Process 1: 47,
Process 1: 48, Process 1: 49, Process 1: 50, Process 1: 51, Process 1: 52, Process 1: 53, Process 1: 54, Process 1: 55, Process 1: 56, Process 1: 57, Process 1: 58, Process
 1: 59, Process 1: 60, Process 1: 61, Process 1: 62, Process 1: 63, Process 1: 64, Process 1: 65, Process 1: 66, Process 1: 67, Process 1: 68, Process 1: 69, Process 1: 70,
 Process 1: 71, Process 1: 72, Process 1: 73, Process 1: 74, Process 1: 75, Process 1: 76, Process 1: 77, Process 1: 78, Process 1: 79, Process 1: 80, Process 1: 81, Proces
s 1: 82, Process 1: 83, Process 1: 84, Process 1: 85, Process 1: 86, Process 1: 87, Process 1: 88, Process 1: 89, Process 1: 90, Process 1: 91, Process 1: 92, Process 1: 93
, Process 1: 94, Process 1: 95, Process 1: 96, Process 1: 97, Process 1: 98, Process 1: 99, Process 1: 100, Process 1: 101, Process 1: 102, Process 1: 103, Process 1: 104,
Process 1: 105, Process 1: 106, Process 1: 107, Process 1: 108, Process 1: 109, Process 1: 110, Process 1: 111, Process 1: 112, Process 1: 113, Process 1: 114, Process 1: 1
15, Process 1: 116, Process 1: 117, Process 1: 118, Process 1: 119, Process 1: 120, Process 1: 121, Process 1: 122, Process 1: 123, Process 1: 124, Process 1: 125, Process
1: 126, Process 1: 127, Process 1: 128, Process 1: 129, Process 1: 130, Process 1: 131, Process 1: 132, Process 1: 133, Process 1: 134, Process 1: 1, Process 0: 70, Process
 0: 71, Process 0: 72, Process 0: 73, Process 0: 74, Process 0: 75, Process 0: 76, Process 0: 77, Process 0: 78, Process 0: 79, Process 0: 80, Process 0: 81, Process 0: 82,
 Process 0: 83, Process 0: 84, Process 0: 85, Process 0: 86, Process 0: 87, Process 0: 88, Process 0: 89, Process 0: 90, Process 0: 91, Process 0: 92, Process 0: 93, Proces
s 0: 94, Process 0: 95, Process 0: 96, Process 0: 97, Process 0: 98, Process 0: 99, Process 0: 100, Process 0: 101, Process 0: 102, Process 0: 103, Process 0: 104, Process
0: 105, Process 0: 106, Process 0: 107, Process 0: 108, Process 0: 109, Process 0: 110, Process 0: 111, Process 0: 112, Process 0: 113, Process 0: 114, Process 0: 115, Proc
ess 0: 116, Process 0: 117, Process 0: 118, Process 0: 119, Process 0: 120, Process 0: 121, Process 0: 122, Process 0: 123, Process 0: 124, Process 0: 125, Process 0: 126,
Process 0: 127, Process 0: 128, Process 0: 129, Process 0: 130, Process 0: 131, Process 0: 132, Process 0: 133, Process 0: 134, Process 0: 135, Process 1: 136, Process 1: 1
```

...

```
Process 0: 344, Process 0: 345, Process 0: 346, Process 0: 347, Process 0: 348, Process 0: 349, Process 0: 350, Process 0: 351, Process 0: 352, Process 0: 353, Process 0: 3
54, Process 0: 355, Process 0: 356, Process 0: 357, Process 0: 358, Process 0: 359, Process 0: 360, Process 0: 361, Process 0: 362, Process 0: 363, Process 0: 364, Process
0: 365, Process 0: 366, Process 0: 367, Process 0: 368, Process 0: 369, Process 0: 370, Process 0: 371, Process 0: 372, Process 0: 373, Process 0: 374, Process 0: 375, Proc
ess 0: 376, Process 0: 377, Process 0: 378, Process 0: 379, Process 0: 380, Process 0: 381, Process 0: 382, Process 0: 383, Process 0: 384, Process 0: 385, Process 0: 386,
Process 0: 387, Process 0: 388, Process 0: 389, Process 0: 390, Process 0: 391, Process 1: 392, Process 1: 393, Process 1: 394, Process 1: 395, Process 1: 396, Process 1: 3
97, Process 1: 398, Process 1: 399, Process 1: 400, Process 1: 401, Process 1: 402, Process 1: 403, Process 1: 404, Process 1: 405, Process 1: 406, Process 1: 407, Process
1: 408, Process 1: 409, Process 1: 410, Process 1: 411, Process 1: 412, Process 1: 413, Process 1: 414, Process 1: 415, Process 1: 416, Process 1: 417, Process 1: 418, Proc
ess 1: 419, Process 1: 420, Process 1: 421, Process 1: 422, Process 1: 423, Process 1: 424, Process 1: 425, Process 1: 426, Process 1: 427, Process 1: 428, Process 1: 429,
Process 1: 430, Process 1: 431, Process 1: 432, Process 1: 433, Process 1: 434, Process 1: 435, Process 1: 436, Process 1: 437, Process 1: 438, Process 1: 439, Process 1: 4
40, Process 1: 441, Process 1: 442, Process 1: 443, Process 1: 444, Process 1: 445, Process 1: 446, Process 1: 447, Process 1: 448, Process 1: 449, Process 1: 450, Process
1: 451, Process 1: 452, Process 1: 453, Process 1: 454, Process 1: 491, Process 0: 392, Process 0: 393, Process 0: 394, Process 0: 395, Process 0: 396, Process 0: 397, Proc
ess 0: 398, Process 0: 399, Process 0: 400, Process 0: 401, Process 0: 402, Process 0: 403, Process 0: 404, Process 0: 405, Process 0: 406, Process 0: 407, Process 0: 408,
Process 0: 409, Process 0: 410, Process 0: 411, Process 0: 412, Process 0: 413, Process 0: 414, Process 0: 415, Process 0: 416, Process 0: 417, Process 0: 418, Process 0: 4
19, Process 0: 420, Process 0: 421, Process 0: 422, Process 0: 423, Process 0: 424, Process 0: 425, Process 0: 426, Process 0: 427, Process 0: 428, Process 0: 429, Process
0: 430, Process 0: 431, Process 0: 432, Process 0: 433, Process 0: 434, Process 0: 435, Process 0: 436, Process 0: 437, Process 0: 438, Process 0: 439, Process 0: 440, Proc
ess 0: 441, Process 0: 442, Process 0: 443, Process 0: 444, Process 0: 445, Process 0: 446, Process 0: 447, Process 0: 448, Process 0: 449, Process 0: 450, Process 0: 451,
Process 0: 452, Process 0: 453, Process 0: 454, Process 0: 455, Process 1: 456, Process 1: 457, Process 1: 458, Process 1: 459, Process 1: 460, Process 1: 461, Process 1: 4
62, Process 1: 463, Process 1: 464, Process 1: 465, Process 1: 466, Process 1: 467, Process 1: 468, Process 1: 469, Process 1: 470, Process 1: 471, Process 1: 472, Process
1: 473, Process 1: 474, Process 1: 475, Process 1: 476, Process 1: 477, Process 1: 478, Process 1: 479, Process 1: 480, Process 1: 481, Process 1: 482, Process 1: 483, Proc
ess 1: 484, Process 1: 485, Process 1: 486, Process 1: 487, Process 1: 488, Process 1: 489, Process 1: 490, Process 1: 491, Process 1: 492, Process 1: 493, Process 1: 494,
Process 1: 495, Process 1: 496, Process 1: 497, Process 1: 498, Process 1: 499, Process 1: 500, 55, Process 0: 456, Process 0: 457, Process 0: 458, Process 0: 459, Process
0: 460, Process 0: 461, Process 0: 462, Process 0: 463, Process 0: 464, Process 0: 465, Process 0: 466, Process 0: 467, Process 0: 468, Process 0: 469, Process 0: 470, Proc
ess 0: 471, Process 0: 472, Process 0: 473, Process 0: 474, Process 0: 475, Process 0: 476, Process 0: 477, Process 0: 478, Process 0: 479, Process 0: 480, Process 0: 481,
Process 0: 482, Process 0: 483, Process 0: 484, Process 0: 485, Process 0: 486, Process 0: 487, Process 0: 488, Process 0: 489, Process 0: 490, Process 0: 491, Process 0: 4
92, Process 0: 493, Process 0: 494, Process 0: 495, Process 0: 496, Process 0: 497, Process 0: 498, Process 0: 499, Process 0: 500,

Final value of shared variable X = 500
```

Μέρος 2

Ερώτημα Α

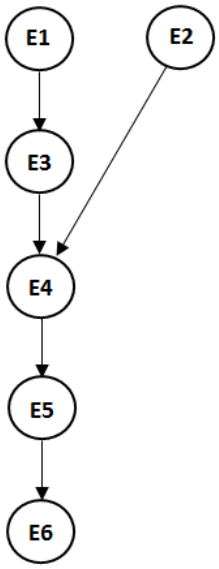
```
E1: A := X;
E2: B := Y;
E3: C := A + Y;
E4: D := C + B;
E5: C := D + B;
E6: D := D + A;
```

Παρατηρώντας προσεκτικά το δοθέν πρόγραμμα καταλήγουμε στα εξής συμπεράσματα:

- Η εντολή Ε3 πρέπει να εκτελεστεί μετά την εντολή Ε1, καθώς το αποτέλεσμά της (τιμή μεταβλητής C) εξαρτάται από την τιμή της μεταβλητής Α η τιμή της οποίας καθορίζεται στην εντολή Ε1.
- Η εντολή Ε2 μπορεί να εκτελεστεί παράλληλα με τις εντολές Ε1 και Ε3 διότι η μεταβλητή Β (Ε2) δεν εξαρτάται από την μεταβλητή Α (Ε1) και η μεταβλητή C (Ε3) δεν εξαρτάται από τη μεταβλητή Β (Ε2).
- Η εντολή Ε4 πρέπει να εκτελεστεί μετά τις εντολές Ε2 και Ε3, καθώς το αποτέλεσμά της (μεταβλητή D) εξαρτάται από τις μεταβλητές Β και C, οι τιμές των οποίων καθορίζονται από τις Ε2 και Ε3 αντίστοιχα.
- Η εντολή Ε5 πρέπει να εκτελεστεί μετά τις εντολές Ε2 και Ε4, καθώς το αποτέλεσμά της (μεταβλητή C) εξαρτάται από τις μεταβλητές Β και D, οι τιμές των οποίων καθορίζονται από τις Ε2 και Ε4 αντίστοιχα. Επιπλέον, θα πρέπει να εκτελεστεί πριν από την Ε6, καθώς σε διαφορετική περίπτωση, η τιμή της μεταβλητής D θα αλλάξει λόγω της Ε6 και συνεπώς η τελική τιμή που θα υπολογιστεί για την μεταβλητή C από την Ε5 θα είναι λανθασμένη.
- Η εντολή Ε6 πρέπει να εκτελεστεί μετά τις εντολές Ε1 και Ε4, καθώς το αποτέλεσμά της (μεταβλητή D) εξαρτάται από τις μεταβλητές Α και D, οι τιμές των οποίων καθορίζονται από τις Ε1 και Ε4 αντίστοιχα.

1)

Με βάση τις παραπάνω παρατηρήσεις καταλήγουμε στο συμπέρασμα πως ο γράφος προτεραιοτήτων με τη μέγιστη δυνατή παραλληλία που αναπαριστά ισοδύναμα την εκτέλεση του παραπάνω ακολουθιακού προγράμματος είναι ο εξής:



2)

```
cobegin
    begin
       E1;
       E3;
    end;
    E2;
coend
E4;
E5;
E6;
```

3)

α) Με πέντε δυαδικούς σημαφόρους:

```
var s1, s2, s3, s4, s5: binary semaphores;
s1 := s2 := s3 := s4 := s5 := 0;
cobegin
    begin E1; up(s1); end
    begin E2; up(s3); end
    begin down(s1); E3; up(s2); end
    begin down(s2); down(s3); E4; up(s4); end
    begin down(s4); E5; up(s5); end
    begin down(s5); E6; end
coend
```

β) Με τρεις δυαδικούς σημαφόρους και έναν γενικό σημαφόρο (αν είχαμε την δυνατότητα να χρησιμοποιήσουμε και έναν γενικό):

```
var s1, s3, s4: binary semaphores;
var s2: semaphore;
s1 := s3 := s4 := 0;
s2 = -1;
cobegin
    begin E1; up(s1); end
    begin E2; up(s2); end
    begin down(s1); E3; up(s2); end
    begin down(s2); E4; up(s3); end
    begin down(s3); E5; up(s4); end
    begin down(s4); E6; end
coend
```

Ερώτημα Β

α) Με πέντε δυαδικούς σημαφόρους:

```
var s1, s2, s3, s4, s5: binary semaphores;
s1 := s2 := s3 := s4 := s5 := 0;
cobegin
    begin E1.1; up(s1); end
    begin down(s3); E1.2; up(s4); end
    begin down(s1); down(s2); E2.1; up(s3); end
    begin down(s4); E2.2; up(s5); end
    begin E3.1; up(s2); end
    begin down(s5); E3.2; end
coend
```

β) Με τρεις δυαδικούς σημαφόρους και έναν γενικό σημαφόρο:

```
var s2, s3, s4: binary semaphores;
var s1: semaphore;
s2 := s3 := s4 := 0;
s1 = -1;
cobegin
    begin E1.1; up(s1); end
    begin down(s2); E1.2; up(s3); end
    begin down(s1); E2.1; up(s2); end
    begin down(s3); E2.2; up(s4); end
    begin E3.1; up(s1); end
    begin down(s4); E3.2; end
coend
```

Ερώτημα Γ

Producer	Consumer	empty	full
		2	0
	wait(full)	2	0
wait(empty)		1	0
signal(full)		1	0
wait(empty)		0	0
	signal(empty)	1	0
signal(full)		1	1
wait(empty)		0	1
signal(full)		0	2
wait(empty)		0	2
	wait(full)	0	1
	signal(empty)	0	1

Ερώτημα Δ

Σημείωση: Στο φυλλάδιο ορίζεται ως Ερώτημα Ε, αλλά λόγω του ότι δεν υπάρχει ερώτημα Δ, θεωρήσαμε ότι έγινε τυπογραφικό λάθος.

Πέντε διεργασίες καταφθάνουν σε ένα υπολογιστικό σύστημα σύμφωνα με τα δεδομένα του ακόλουθου πίνακα:

Όνομα Διεργασίας	Χρονική Στιγμή Άφιξης	Απαιτήσεις Χρόνου Εκτέλεσης	Προτεραιότητα
P_1	0	11	1
P_2	2	4	4
P_3	3	9	5
P_4	3	12	3
P_5	5	5	2

Παραδοχές (από εκφώνηση):

1. Ο χρόνος εναλλαγής (context switch) είναι αμελητέος.
2. Για τον αλγόριθμο RR θεωρούμε πως αν τη χρονική στιγμή που μια νέα διεργασία έρχεται στο σύστημα διακόπτεται η εκτέλεση μιας διεργασίας (γιατί τελειώνει το κβάντο χρόνου της), τότε η νέα διεργασία εισέρχεται πριν από αυτήν που διακόπτεται στην ουρά των έτοιμων διεργασιών.

Επιπλέον παραδοχές:

Στον αλγόριθμο RR έγινε η παραδοχή ότι σε περίπτωση που μία διεργασία, έστω Α, καταφθάσει το σύστημα την χρονική στιγμή t_1 και την ίδια χρονική στιγμή τελειώσει μία άλλη διεργασία, έστω Β, το κβάντο της στην CPU, τότε πρώτα μπαίνει στην ουρά των έτοιμων διεργασιών η διεργασία Α και μετά η διεργασία Β.

Στα διαγράμματα Gantt που παρατίθενται στη συνέχεια για κάθε αλγόριθμο, το **γκρι** χρώμα αντιστοιχεί σε χρήση της CPU από τη διεργασία P_1, το **κίτρινο** σε χρήση της CPU από τη διεργασία P_2, το **κόκκινο** χρήση της CPU από τη διεργασία P_3, το **καφέ** σε χρήση της CPU από τη διεργασία P_4 και, τέλος, το **μπλε** σε χρήση της CPU από τη διεργασία P_5.

ΧΔ: Χρόνος Διεκπεραίωσης

ΧΑ: Χρόνος Αναμονής

ΜΧΔ: Μέσος Χρόνος Διεκπεραίωσης

ΜΧΑ: Μέσος Χρόνος Αναμονής

Οι χρόνοι διεκπεραίωσης και αναμονής για κάθε διεργασία υπολογίζονται ως εξής:

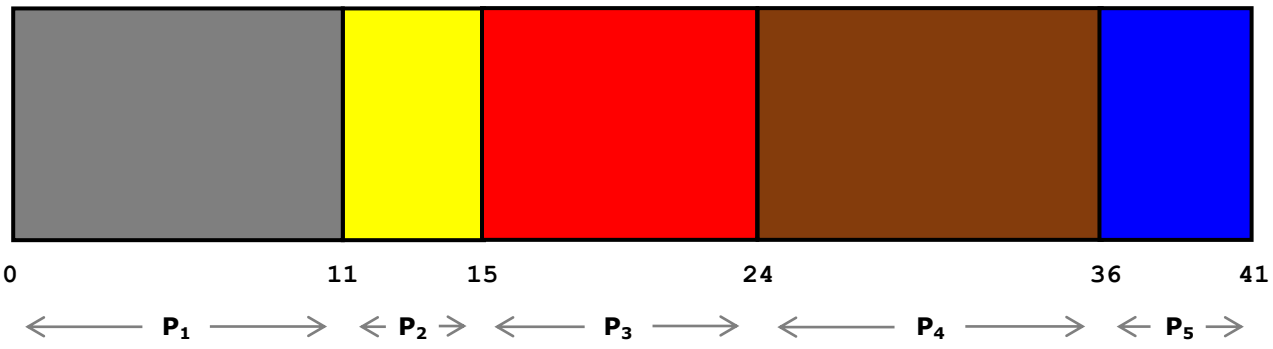
Αν t_1 είναι ο χρόνος εισόδου (χρονική στιγμή άφιξης) και t_2 ο χρόνος εξόδου (χρονική στιγμή ολοκλήρωσης), τότε ο χρόνος διεκπεραίωσης είναι: ΧΔ = t_2 - t_1.

Εάν t_1 είναι η χρονική στιγμή εισόδου και t_2 η χρονική στιγμή εξόδου και t_{CPU} είναι ο χρόνος που χρειάζεται η διεργασία για εξυπηρέτηση από τη CPU (απαιτήσεις χρόνου εκτέλεσης) τότε ο χρόνος αναμονής ισούται με: ΧΑ = t_2 - t_1 - t_{CPU} = ΧΔ - t_{CPU}.

<u>Σημείωση:</u> Θεωρούμε πως όσο μεγαλύτερη είναι η τιμή της προτεραιότητας τόσο πιο ισχυρή (μεγάλη) είναι η προτεραιότητα.

FCFS (First Come First Served)

Διάγραμμα Gantt:



Η εκτέλεση των διεργασιών θα ολοκληρωθεί την 41η χρονική μονάδα (δεν αναφέρεται στην εκφώνηση η χρονική μονάδα μέτρησης, πχ. ms).

$ΧΔ_{P1}$ = 11 – 0 = 11

$ΧΔ_{P2}$ = 15 - 2 = 13

$ΧΔ_{P3}$ = 24 – 3 = 21

$ΧΔ_{P4}$ = 36 – 3 = 33

$ΧΔ_{P5}$ = 41 – 5 = 36

Συνεπώς, ο μέσος χρόνος διεκπεραίωσης ισούται με:

ΜΧΔ = (11 + 13 + 21 + 33 + 36) / 5 = 22,8.

$ΧΑ_{P1}$ = 11 – 0 - 11 = 0

$ΧΑ_{P2}$ = 15 – 2 - 4 = 9

$ΧΑ_{P3}$ = 24 – 3 - 9 = 12

$ΧΑ_{P4}$ = 36 – 3 - 12 = 21

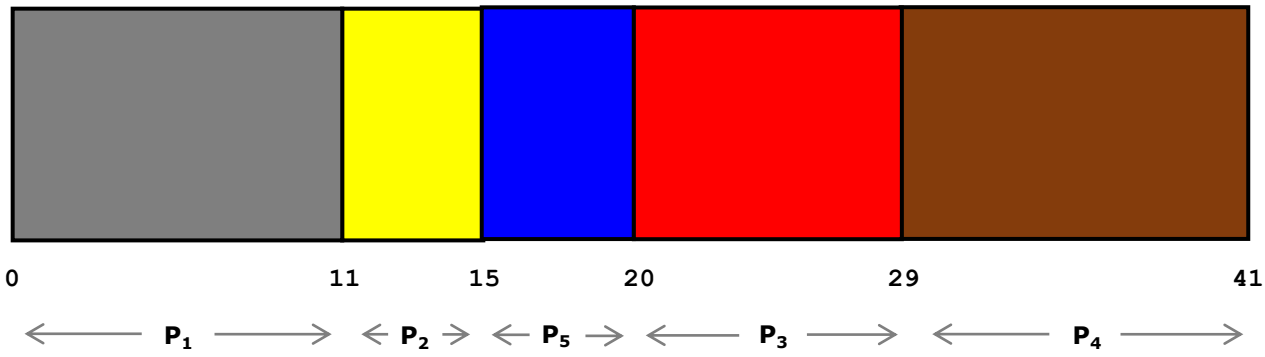
$ΧΑ_{P5}$ = 41 – 5 - 5 = 31

Επομένως, ο μέσος χρόνος αναμονής ισούται με:

ΜΧΑ = (0 + 9 + 12 + 21 + 31) / 5 = 14,6.

SJF (Shortest Job First)

Διάγραμμα Gantt:



$ΧΔ_{P1}$ = 11 – 0 = 11

$ΧΔ_{P2}$ = 15 – 2 = 13

$ΧΔ_{P3}$ = 29 – 3 = 26

$ΧΔ_{P4}$ = 41 – 3 = 38

$ΧΔ_{P5}$ = 20 – 5 = 15

Άρα, ο μέσος χρόνος διεκπεραίωσης ισούται με:

ΜΧΔ = (11 + 13 + 26 + 38 + 15) / 5 = 20,6.

$ΧΑ_{P1}$ = 11 – 0 - 11 = 0

$ΧΑ_{P2}$ = 15 – 2 - 4 = 9

$ΧΑ_{P3}$ = 29 – 3 - 9 = 17

$ΧΑ_{P4}$ = 41 – 3 – 12 = 26

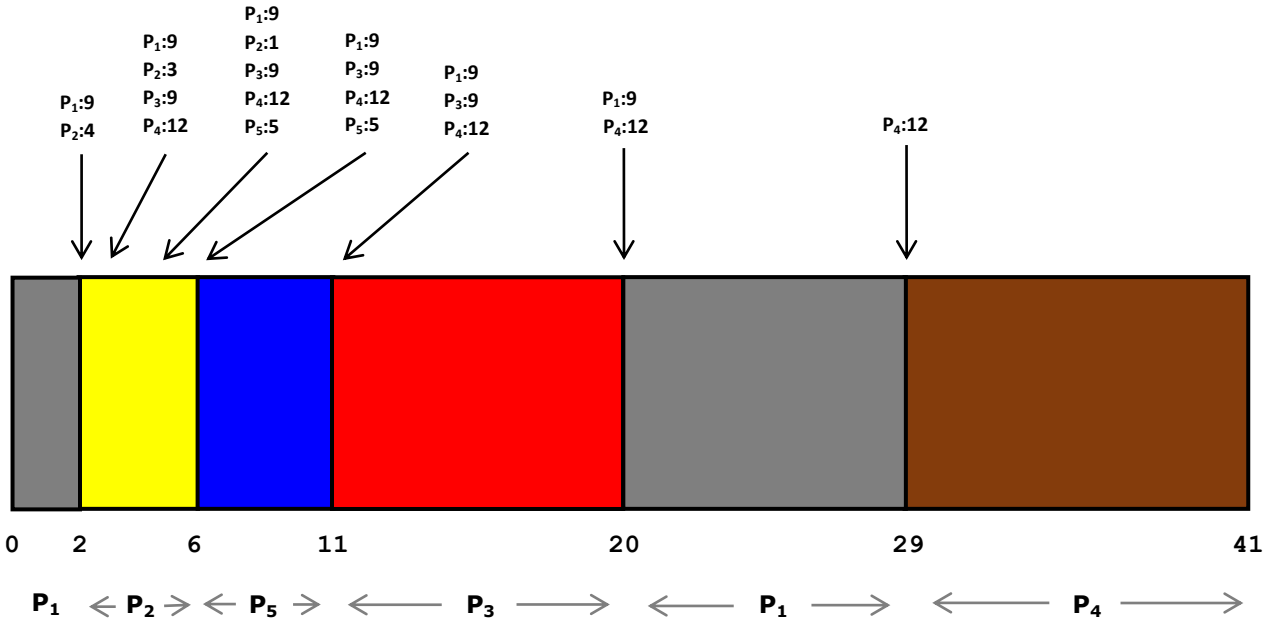
$ΧΑ_{P5}$ = 20 – 5 - 5 = 10

Συνεπώς, ο μέσος χρόνος αναμονής ισούται με:

ΜΧΑ = (0 + 9 + 17 + 26 + 10) / 5 = 12,4.

SRTF (Shortest Remaining Time First)

Διάγραμμα Gantt:



$ΧΔ_{P1}$ = 29 – 0 = 29

$ΧΔ_{P2}$ = 6 - 2 = 4

$ΧΔ_{P3}$ = 20 – 3 = 17

$ΧΔ_{P4}$ = 41 – 3 = 38

$ΧΔ_{P5}$ = 11 – 5 = 6

Επομένως, ο μέσος χρόνος διεκπεραίωσης ισούται με:

ΜΧΔ = (29 + 4 + 17 + 38 + 6) / 5 = 18,8.

$ΧΑ_{P1}$ = 29 – 0 - 11 = 18

$ΧΑ_{P2}$ = 6 – 2 - 4 = 0

$ΧΑ_{P3}$ = 20 – 3 - 9 = 8

$ΧΑ_{P4}$ = 41 – 3 – 12 = 26

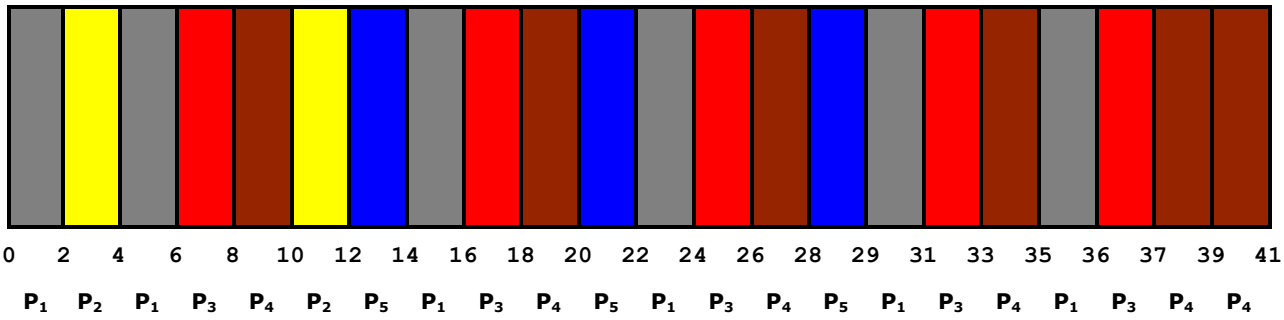
$ΧΑ_{P5}$ = 11 – 5 - 5 = 1

Άρα, ο μέσος χρόνος αναμονής ισούται με:

ΜΧΑ = (18 + 0 + 8 + 26 + 1) / 5 = 10,6.

RR (Round Robin) με κβάντο χρόνου 2 χρονικές μονάδες

Διάγραμμα Gantt:



$ΧΔ_{P1}$ = 36 – 0 = 36

$ΧΔ_{P2}$ = 12 - 2 = 10

$ΧΔ_{P3}$ = 37 – 3 = 34

$ΧΔ_{P4}$ = 41 – 3 = 38

$ΧΔ_{P5}$ = 29 – 5 = 24

Συνεπώς, ο μέσος χρόνος διεκπεραίωσης ισούται με:

ΜΧΔ = (36 + 10 + 34 + 38 + 24) / 5 = 28,4.

$ΧΑ_{P1}$ = 36 – 0 - 11 = 25

$ΧΑ_{P2}$ = 12 – 2 - 4 = 6

$ΧΑ_{P3}$ = 37 – 3 - 9 = 25

$ΧΑ_{P4}$ = 41 – 3 – 12 = 26

$ΧΑ_{P5}$ = 29 – 5 - 5 = 19

Επομένως, ο μέσος χρόνος αναμονής ισούται με:

ΜΧΑ = (25 + 6 + 25 + 26 + 19) / 5 = 20,2.

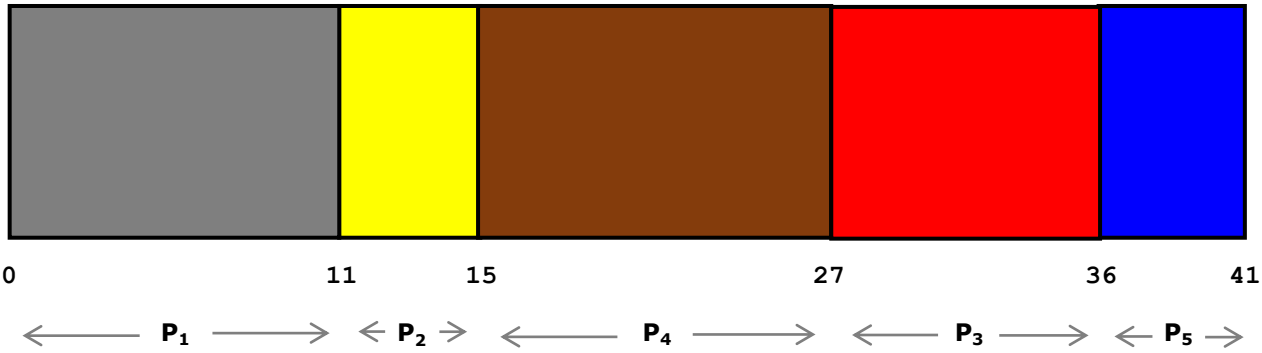
Εάν θεωρήσουμε ότι η προτεραιότητα με τιμή 1 είναι πιο ισχυρή από την προτεραιότητα με τιμή 2, που αντίστοιχα είναι πιο ισχυρή από την προτεραιότητα με τιμή 3 κ.ο.κ., η σειρά εκτέλεσης και ολοκλήρωσης των διεργασιών στους αλγόριθμους FCFS, SRTF και RR μεταβάλλεται.

Αντιθέτως, η σειρά εκτέλεσης και ολοκλήρωσης των πέντε διεργασιών παραμένει η ίδια για τον αλγόριθμο SJF ανεξαρτήτως της όποιας παραδοχής κάνουμε για την προτεραιότητα.

Στη συνέχεια παρατίθενται τα διαγράμματα Gantt και υπολογίζονται οι ζητούμενοι χρόνοι για τους αλγόριθμους FCFS, SRTF και RR για την περίπτωση που η προτεραιότητα με τιμή 1 είναι πιο ισχυρή από την προτεραιότητα με τιμή 2 κ.ο.κ.

FCFS (First Come First Served)

Διάγραμμα Gantt:



$XΔ_{P1}$ = 11 – 0 = 11

$XΔ_{P2}$ = 15 - 2 = 13

$XΔ_{P3}$ = 36 – 3 = 33

$XΔ_{P4}$ = 27 – 3 = 24

$XΔ_{P5}$ = 41 – 5 = 36

Συνεπώς ο μέσος χρόνος διεκπεραίωσης ισούται με:

ΜΧΔ = (11 + 13 + 33 + 24 + 36) / 5 = 23,4

XA_{P1} = 11 – 0 - 11 = 0

XA_{P2} = 15 – 2 - 4 = 9

XA_{P3} = 36 – 3 - 9 = 24

XA_{P4} = 27 – 3 - 12 = 12

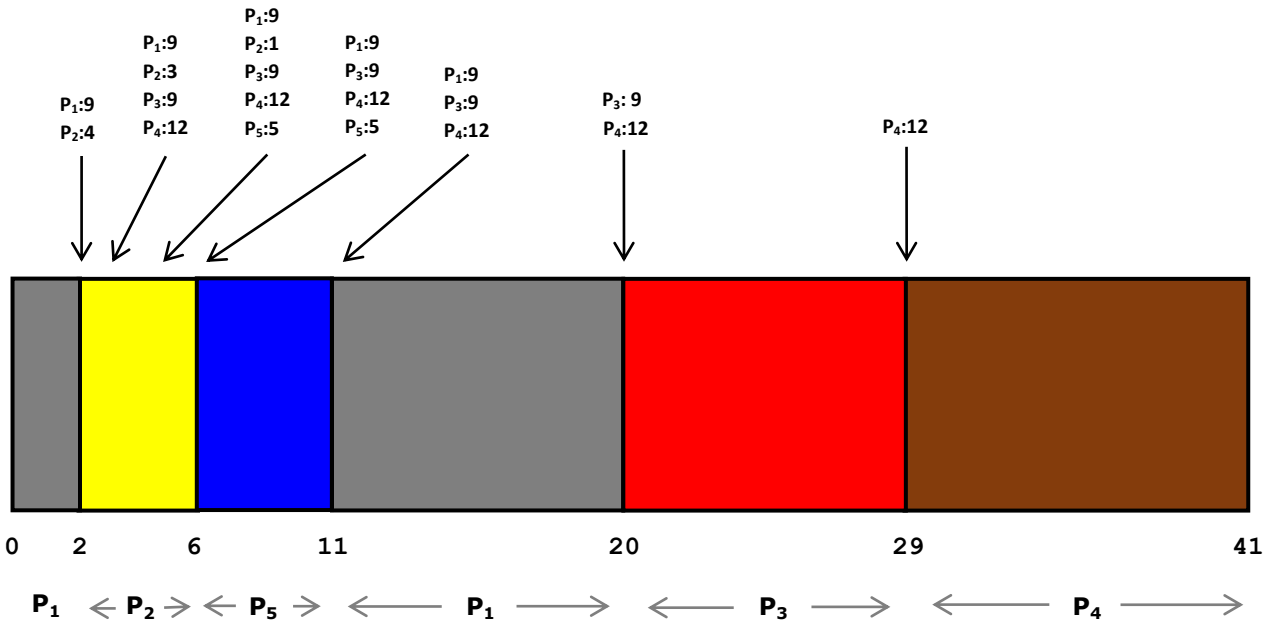
XA_{P5} = 41 – 5 - 5 = 31

Άρα, ο μέσος χρόνος αναμονής ισούται με:

ΜΧΑ = (0 + 9 + 24 + 12 + 31) / 5 = 15,2.

SRTF (Shortest Remaining Time First)

Διάγραμμα Gantt:



$XΔ_{P1}$ = 20 – 0 = 20

$XΔ_{P2}$ = 6 – 2 = 4

$XΔ_{P3}$ = 29 – 3 = 26

$XΔ_{P4}$ = 41 – 3 = 38

$XΔ_{P5}$ = 11 – 5 = 6

Συνεπώς, ο μέσος χρόνος διεκπεραίωσης ισούται με:

ΜΧΔ = (20 + 4 + 26 + 38 + 6) / 5 = 18,8.

XA_{P1} = 20 – 0 – 11 = 9

XA_{P2} = 6 – 2 – 4 = 0

XA_{P3} = 29 – 3 – 9 = 17

XA_{P4} = 41 – 3 – 12 = 26

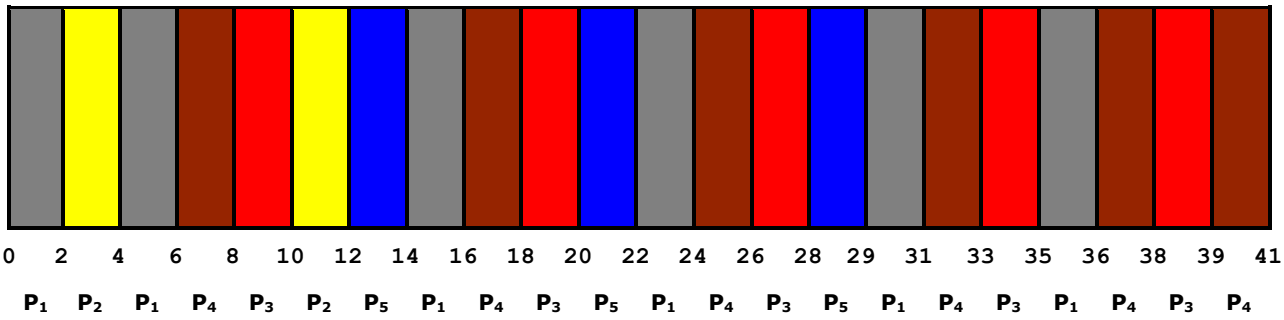
XA_{P5} = 11 – 5 – 5 = 1

Επομένως, ο μέσος χρόνος αναμονής ισούται με:

ΜΧΑ = (9 + 0 + 17 + 26 + 1) / 5 = 10,6.

RR (Round Robin) με κβάντο χρόνου 2 χρονικές μονάδες

Διάγραμμα Gantt:



$ΧΔ_{P1}$ = 36 – 0 = 36

$ΧΔ_{P2}$ = 12 - 2 = 10

$ΧΔ_{P3}$ = 39 – 3 = 36

$ΧΔ_{P4}$ = 41 – 3 = 38

$ΧΔ_{P5}$ = 29 – 5 = 24

Συνεπώς, ο μέσος χρόνος διεκπεραίωσης ισούται με:

ΜΧΔ = (36 + 10 + 36 + 38 + 24) / 5 = 28,8.

$ΧΑ_{P1}$ = 36 – 0 - 11 = 25

$ΧΑ_{P2}$ = 12 – 2 - 4 = 6

$ΧΑ_{P3}$ = 39 – 3 - 9 = 27

$ΧΑ_{P4}$ = 41 – 3 – 12 = 26

$ΧΑ_{P5}$ = 29 – 5 - 5 = 19

Επομένως, ο μέσος χρόνος αναμονής ισούται με:

ΜΧΑ = (25 + 6 + 27 + 26 + 19) / 5 = 20,6.