**ΛΕΙΤΟΥΡΓΙΚΑ ΣΥΣΤΗΜΑΤΑ**

Ονοματεπώνυμο: Παναγιώτης Ντενέζος

Α.Μ.: 5853

email: [ntenezos@ceid.upatras.gr](mailto:ntenezos@ceid.upatras.gr)

Όλα τα ερωτήματα της άσκησης υλοποιήθηκαν και πέρασαν με επιτυχία τον αυτόματο έλεγχο.

* **syn\_process\_1**

Στο συγκεκριμένο πρόγραμμα δημιουργούνται δυο διεργασίες με χρήση της συνάρτησης *fork().* Και η διεργασία πατέρας όσο και η διεργασία παιδί πρέπει να χρησιμοποιήσουν τη συνάρτηση *display()* για να εμφανίσουν το καθένα το δικό του μήνυμα. Για να αποφευχθεί το “ανακάτεμα” των μηνυμάτων χρησιμοποιούνται σημαφόροι. Πιο συγκεκριμένα, όταν μια από της δυο διεργασίες χρησιμοποιούσε την συνάρτηση *display()* έμπαινε σε κρίσιμη περιοχή και η άλλη διεργασία δεν μπορούσε να την καλέσει μέχρι να τελειώσει η προηγουμένη. Ο έλεγχος για την κρίσιμη περιοχή γίνεται με χρήση σημαφόρων. Σε κάθε διεργασία, πρώτα η τιμή της σημαφόρου γίνεται 0, ώστε η κλίση της συνάρτησης *display()* να μπεί στην κρίσιμη περιοχή μέχρι να ολοκληρωθεί η λειτουργία της και μετά η τιμή της σημαφόρου γίνεται 1, ώστε να βγεί από την κρίσιμη περιοχή και να αφήσει “ελεύθερη“ την συνάρτηση *display()* για την όποια διεργασία θέλει να την χρησιμοποιήσει.

* **syn\_process\_2**

Στο συγκεκριμένο πρόγραμμα δημιουργούνται πάλι δυο διεργασίες με χρήση της συνάρτησης *fork().* Όπως και στο προηγούμενο ερώτημα και οι δυο θέλουν να κάνουν χρήση της συνάρτησης *display()* για να εμφανίσουν το καθένα το δικό του μήνυμα. Για να ακολουθηθεί η σειρά που ζητείτε «abcd», γίνεται χρήση σημαφόρων. Πιο συγκεκριμένα, επειδή πρέπει η μια διεργασία να ακολουθάει την άλλη γίνεται χρήση μιας κοινής μεταβλητής, η οποία κοιμίζει και ξυπνάει αντίστοιχα τις διεργασίες. Για να επιτευχθεί αυτό, καθώς η συνάρτηση *fork()* δημιουργεί αντίγραφο των μεταβλητών για την διεργασία παιδί, γίνεται χρήση της κοινής μνήμης. Η μεταβλητή που υπάρχει στην κοινή μνήμη είναι αυτή που κοιμίζει και ξυπνάει αντίστοιχα τις δυο διεργασίες.

* **syn\_thread\_1**

Στο συγκεκριμένο πρόγραμμα δημιουργούνται δυο νήματα. Όπως στο πρόγραμμα syn\_process\_1, πρέπει να εμφανιστούν τα μηνύματα και των δυο νημάτων ολόκληρα και χωρίς να “ανακατευτούν”. Για να επιτευχθεί λοιπόν αυτό χρησιμοποιείται ο αμοιβαίος αποκλεισμός. Πιο συγκεκριμένα, όταν ένα νήμα εκτελείται, για όσο κάνει χρήση της συνάρτησης *display()* κλειδώνει και δεν μπορεί άλλο νήμα να κάνει χρήση της ίδιας συνάρτησης.

* **syn\_thread\_2**

Στο συγκεκριμένο πρόγραμμα δημιουργούνται πάλι δυο νήματα. Όπως στο πρόγραμμα syn\_process\_2, πρέπει να εμφανιστούν τα μηνύματα και των δυο νημάτων με την ακόλουθη σειρά «abcd». Για να ακολουθηθεί η σειρά που ζητείτε, γίνεται χρήση μια κοινής μεταβλητής με την ίδια λογική με το πρόγραμμα syn\_process\_2, δηλαδή μόλις ολοκληρωθεί το μήνυμα που πρέπει να εμφανιστεί καλεί το άλλο νήμα να τρέξει.

* **Προβλήματα που υπήρξαν**

Τα προβλήματα που υπήρξαν κατά την υλοποίηση των παραπάνω προγραμμάτων, ήταν αρχικά στο 1ο προγραμμα η χρήση των up και down για την υλοποίηση του ελέγχου της κρίσιμης περιοχής, το οποίο όμως λύθηκε αρκετά γρήγορα. Στην πορεία ένα ακόμα πρόβλημα που εμφανίστηκε ήταν η χρήση της κοινής μεταβλητής στο 2ο πρόγραμμα, το οποίο ήταν περισσότερο θέμα θεωρίας, συνεπώς με πιο προσεχτικό διάβασμα της αντίστοιχης θεωρίας το πρόβλημα λύθηκε. Μια άλλη υποψήφια υλοποίηση για το συγκεκριμένο ερώτημα ήταν με χρήση 2 σημαφόρων. Δεν έγινε όμως κάποια προσπάθεια υλοποίησης της. Τέλος το τελευταίο πρόβλημα που παρατηρήθηκε ήταν το κλείδωμα των δυο συναρτήσεων όταν καλούσαν την συνάρτηση *display()*. Το πρόβλημα λύθηκε, καθώς όταν κάποιο νήμα καλεί την συνάρτηση του το κλείδωμα γίνεται μόνο στα conditions.