Λειτουργικά Συστήματα – Εργασία 1
Μυλωνόπουλος Δημήτριος – Α.Μ: 1115201600112

Στην εργασία αυτή μας ζητήθηκε να υλοποιήσουμε ένα πρόγραμμα προσομοίωσης γραμμής παραγωγής 4 σταδίων. Τα στάδια είναι τα εξής: Το στάδιο της κατασκευής, το στάδιο του βαψίματος, το στάδιο του ελέγχου και τέλος της συναρμολόγησης του τελικού προιόντος. Για την υλοποίηση των διαφόρων σταδίων κατασκευής χρησιμοποίηθηκαν διεργασίες και συγκεκριμένα 8. 3 για το στάδιο κατασκευής δεδομένου ότι είχαμε 3 διαφορετικά είδη εξαρτημάτων, 1 για το βαφείο, 3 για τους ελέγχους (κάθε μια για διαφορετικό είδος εξαρτμήματος) και τέλος 1 για το στάδιο της συναρμολόγησης. Για την μεταφορά των εξαρτημάτων μεταξύ των διεργασιών ήταν απαραίτητη η χρήση διαμοιραζόμενης μνήμης. Συγκεκριμένα χρησιμοποιείται διαμοιραζόμενη μνήμη μεταξύ των τμημάτων κατασκευής και του βαφείου η οποία αποθηκεύει ένα Εξάρτημα μέσα της, άλλες τρεις μεταξύ του βαφείου και των τμημάτων ελέγχου, επίσης μεγέθους ενός εξαρτήματος, και τέλος μια ακόμη μεταξύ των τμημάτων ελέγχου και του τμήματος συναρμολόγησης. Καταλαβαίνουμε πως για μια τέτοια προσωμοίωση στην οποία χρησιμοποιήθηκε γενικά λίγη διαμοιραζόμενη μνήμη απαραίτητος είναι ο κατάλληλος συγχρονισμός των διεργασιών. Για να επιτευχθεί συγχρονισμός χρησιμοποιήθηκε ένα σύνολο σημαφόρων. Όλα τα παραπάνω θα αναλυθούν στη συνέχεια.

Κατ’ αρχάς για την δημιουργία ανάκτηση και διαγραφή σημαφόρων και διαμοιραζόμενης μνήμης χρησιμοποιήθηκαν οι κώδικες του εργαστηρίου, οι οποίοι υλοποιήθηκαν σε συναρτήσεις για εξοικονόμηση χώρου και για να είναι το πρόγραμμα πιο ευανάγνωστο. Τους κωδικές αυτούς μπορείτε να τους βρείτε στο **file_shared_sem_operations.c.** Για τους σημαφόρους και τη διαμοιραζόμενη μνήμη χρησιμοποιούνται αρχεία **.key** που αποθηκεύονται τα κλειδιά τους έτσι ώστε οι διεργασίες να μπορούν να τα ανακτούν γνωρίζοντας το αρχείο κλειδί. Επιπλεόν για τις πράξεις στους σημαφόρους όπως up(), down(), και ανάθεση τιμής, υλοποιήθηκαν συναρτήσεις, οι οποίες βασίστηκαν στους κώδικες του εργαστηρίου με μερικές τροποποιήσεις. Τις συναρτήσεις αυτές μπορείτε να τις βρείτε στο **sem_operations.c.**

Θα προσεγγίσουμε το πρόγραμμα “από έξω προς τα μέσα” δίνοντας μια γενική ιδέα στην αρχή και στη συνέχεια εμβαθύνοντας σε λεπτομέριες. Στη main συνάρτηση μου ουσιαστικά δεσμεύεται η απαρραίτητη μνήμη και σημαφόροι που χρειάζονται και στην συνέχεια αρχικοποιώ τους σημαφόρους με τις κατάλληλες τιμές. Έπειτα με 8 fork() δημιουργούνται 8 διεργασίες παιδιά στις οποίες στην κάθε μια καλείται η συνάρτηση με τα κατάλληλα ορίσματα που αντιστοιχεί στην κάθε διεργασία. Μόλις η διεργασία τελειώσει καλείται exit() για να βγει το πρόγραμμα απ τη διεργασία του παιδιού. Συγκεκριμένα οι συναρτήσεις που χρησιμοποιήθηκαν για την κάθε διεργασία είναι οι **Construction(), Painting(), QualityTesting() και Assembly()** που υλοποιήθηκαν στο **Processes.c,** των οποίων τη λειτουργία, τους σημαφόρους και τη διαμοιραζόμενη μνήμη που χρησιμοποίησαν θα αναλύσουμε στη συνέχεια. Η πατρική διεργασία περιμένει τα παιδιά της να κανουν exit() με μια λούπα απο 8 συνολικά waits. Έτσι μόλις κάνουν exit() και τα 8 παιδιά η πατρική διεργασία προχωρά στη διαγραφή των σημαφόρων και της διαμοιραζόμενης μνήμης που δεσμεύτηκαν για το πρόγραμμα.

Ας αναλύσουμε τώρα τη λειτουργία της κάθε συνάρτησης του **Processes.h:**

1. Η **Contruction()** παίρνει ως ορίσματα των αριθμό των εξαρτημάτων που απαιτείται να δημιουργήσει, όπως και το είδος αυτών των εξαρτημάτων (τα είδη παίρνουν τις τιμές 0-2). Για τη συνάρτηση αυτή ανακτούμε και κάνουμε attach τη **διαμοιραζόμενη μνήμη μεταξύ του τμήματος κατασκευής και του βαφείου** όπως και ένα **σετ 3ων available-empty-full σημαφόρων** για το συγχρονισμό των 3ων διεργασιών κατασκευής και του βαφείου. Για το συγχρονισμό τους χρησιμοποιήθηκε η γνωστή χρήση σημαφόρων του προβλήματος

συγχρονισμού **Consumer-Producer** ή αλλιώς **Bounded Buffer Problem.** Οι τιμές των 3ων σημαφόρων για το πρόβλημα αρχικοποιούνται ως εξής. Ο available με την τιμή 1 δεδομένου ότι στην αρχή η διαμοιραζόμενη δε χρησιμοποιείται από κάποια διεργασία. Ο empty με την τιμή 1 δεδομένου οτι η διαμοιραζόμενη στην αρχή είναι άδεια και αντίστοιχα ο full με 0. Μέσα στο πρόγραμμα έχουμε μια λούπα με iterations ίσες με τον αριθμό των εξαρτημάτων που θέλουμε να φτιαχτούν. Στην αρχή της λούπας δημιουργείται ένα εξάρτημα με την κλήση της **CreatePart()** με **τυχαίο 4ψήφιο ID** και τον τύπο που έχουμε ορίσει και χρονοσημαίνεται η αρχή της δημιουργίας του. Έπειτα εμείς θέλουμε αυτό να το προσθέσουμε στη διαμοιραζόμενη για το διαβάσει το Βαφείο. Γι αυτό κατεβάζουμε τον empty σημαφόρο και μετά κατεβάζουμε τον available επειδή θέλουμε να καταλάβουμε τη διαμοιραζόμενη μνήμη. Όταν δωθεί πρόσβαση γράφουμε το εξάρτημα στη διαμοιραζόμενη και μετά ξανανέβαζουμε το available και ανεβάζουμε το full. Έτσι καμια άλλη διεργασία δε θα μπορεί να γράψει στη μνήμη μέχρι να διαβάσει το εξάρτημα το βαφείο. Αποσυνάπτουμε τη διαμοιραζόμενη μνήμη μόλις τελειώσει η κατασκευή όλων των εξαρτημάτων και φεύγουμε απ τη διεργασία.

2. Η **Painting()** παίρνει ως όρισμα τον αριθμό των εξαρτημάτων που πρέπει να βάψει. Για το κομμάτι επικοινωνίας της με τη διαμοιραζόμενη μνήμη με το τομέα κατασκευής, παίρνει τους αντίστοιχους σημαφόρους και διαμοιραζόμενη μνήμη και κάνει τα εξής. Κάνει down() τον full και down() τον available για να ανακτήσει πρόσβαση στο εξάρτημα της διαμοιραζόμενης και προφανώς όταν τελειώσει με την ανάκτηση του εξαρτήματος κάνει up() τους available και empty. Μόλις το ανακτήσει χρονοσημαίνει τον χρόνο εισαγωγής του εξαρτήματος στο βαφείο και δαπανά χρόνο ανάλογο του είδους του εξαρτήματος μέσω usleep() για το βάψιμο του. Μόλις τελειώσει το βάψιμο πρέπει να το προωθήσει στη **διαμοιραζόμενη μνήμη μεταξύ βαφείου και σταδίων ελέγχου**. Για τον συγχρονισμό των διεργασιών αυτών γύρω απ τη διαμοιραζόμενη μνήμη, χρησιμοποιήθηκαν 3 σετ σημαφόρων available-full-empty ένα set για καθε είδος τμήματος ελέγχου. Κάθε φορά αναλόγως το είδος του εξαρτήματος στέλνουμε με την ίδια διαδικασία που περιγράψαμε και παραπάνω για το **Construction(),** το εξάρτημα στο κατάλληλο shared memory για να το λάβει το QualityTesting. Τέλος κάνω detatch τις διαμοιραζομενες μνήμες.
3. Η **QualityTesting()** παίρνει ως ορίσμα τον αριθμό των εξαρτημάτων που πρέπει να ελένξει και τον τύπο των εξαρτημάτων αυτών. Αρχικά κάνουμε επισύναψη στην κατάλληλη διαμοιραζόμενη μνήμη ανάμεσα στο Βαφείο και το Τμήμα ελέγχου, δηλαδή σε αυτή που αντιστοιχεί στον τύπο εξαρτημάτων που ελέγχει. Έπειτα ζητάμε να δεσμεύσουμε τη διαμοιραζόμενη για να πάρουμε το στοιχείο, με τον τρόπο που έχουμε περιγράψει παραπάνω **(down(full)down(available) //read// up(available) up(empty))** και επειτα κάνουμε usleep για χρόνο ανάλογο του είδους του εξαρτήματος. Μετά θέλουμε να το προωθήσουμε στο κομμάτι της συναρμολόγησης γι αυτό και κάνουμε στους σημαφόρους που αντιστοιχούν στο shared memory μεταξύ ελέγχου και συναρμολόγησης **(down(empty)down(available)//write//up(available)up(full)).** Και τέλος αποσυνάπτουμε τις διαμοιραζόμενες μνήμες απ' τη διεργασία όταν αυτή τελειώνει.
4. Τέλος φτάνουμε στο κομμάτι της συναρμολόγησης, δηλαδή στην συνάρτηση **Assembly()** με ορίσμα τον αριθμό των εξαρτημάτων που θα λάβει. Εδώ με το ίδιο ακριβώς σκεπτικό που είχαμε και στα παραπάνω για το read διαβάζω το εξάρτημα και το βάζω σε μια απ' τις 3 ούρες που του αντιστοιχεί, δηλαδή σε αυτή που είναι για τον τύπο του εξαρτήματος. Οι ουρές μας είναι δυναμικές. Αν και οι τρείς ουρές περιέχουν στοιχεία τότε αυτά τα στοιχεία τα κάνω DeQueue και τα περνάω ως όρισμα στη **CreateFinalProduct()** όπου και το τελικό προιόν χρονοσημαίνεται με τη στιγμή της δημιουργίας του. Επίσης στη **CreateFinalProduct()** δίνουμε και μια άλλη τιμή στο τελικό προϊόν που είναι η χρονοσήμανση του εξαρτήματος του που δημιουργήθηκε πρώτο. Ακριβώς πριν το **CreateFinalProduct()** δαπανάται μέσω usleep σταθερός χρόνος για τη δημιουργία του. Για τις χρονοσημάνσεις των εξαρτημάτων και των

τελικών προιόντων χρησιμοποιήθηκε το έτοιμο struct της c το **timespec** που εκεί αποθηκεύεται το αποτέλεσμα της clock_gettime(CLOCK_MONOTONIC, struct timespect *result). Συγκεκριμενα χρησιμοποιηθηκε το CLOCK_MONOTONIC για να μην έχουμε θέματα σε περίπτωση που αλλάξει η ώρα κατά τη διάρκεια της δημιουργιας ενός προιόντος. Για τον υπολογισμό των μέσων όρων που ζητήθηκε να εκτυπωθούν στο τέλος η υλοποιήση έχει ως εξής. Για τον μέσο χρόνο αναμονής ενός εξαρτήματος για την εισαγωγή του στο βαφείο βρέθηκε η διαφορά της χρονοσήμανσης της εισαγωγής του στο βαφείο και της χρονοσήμανσης της δημιουργιας του. Το αποτέλεσμα το κρατήσαμε σε δευτερόλεπτα μετατρέποντας κατάλληλα τα nsec του timespec σε δευτερόλεπτα. Υπολογίστηκε το άθροισμα των διαφορών όλων των εξαρτημάτων και το αποτέλεσμα το διαιρέσαμε με τον αριθμό των εξαρτημάτων για να βρεθεί ο μέσος χρόνος. Για τον υπολογισμό του μέσου χρόνου απ' την κατασκευή του πρώτου εξαρτήματος του τελικού προϊόντος μέχρι την κατασκευή του προιόντος βρίσκουμε την διαφορά των δύο αντίστοιχων χρονοσημάνσεων που βρίσκονται μέσα στη δομή του τελικού προϊόντος και τη μετατρέπουμε σε δευτερόλεπτα. Το άθροισμα των διαφορών δια των συνολικό αριθμό των τελικών προϊόντων (δηλαδή το 1/3 του συνολικού αριθμού εξαρτημάτων) είναι ο μέσος χρόνος που ζητήθηκε. Τέλος αποσυνάπτουμε τη διαμοιραζόμενη μνήμη απ τη διεργασία.

Για να κάνετε το πρόγραμμα compile πηγαίνεται στο φάκελο του πηγαίου κώδικα και εκτελέστε την εντολή make. Για να εκτελέσετε το πρόγραμμα εκτελέστε στο working directory την εντολή ./Simulation <arg: number of Components>. Για να καθαρίσετε τα .key τα .o και το εκτελέσιμο εκτελέστε την εντολή make clean.

Σχετικά με τις μετρήσεις του προγράμματος:
Το πρόγραμμα από default έχει της τάξης του 2ms καθυστέρηση στα usleep. Δηλαδή ο χρόνος αναμονής στο βαφείο είναι (type+1) * 2ms εφόσον το type ξεκινά απο 0, στο κομμάτι του ελέγχου (3-type) * 2ms και η συναρμολόγηση παίρνει σταθερό χρόνο 2ms. Για να το αλλάξετε θέστε διαφορετική σταθερά στο **#define** του **Processes.h.** Στα linux της σχολής τα αποτελέσματα για αυτή την καθυστέρηση είναι:
Για 10 εξαρτήματα:
Average waiting time from production to entering the Painting department: 0.015769 seconds
Average production time of Final Product: 0.045024 seconds

Για 100 εξαρτήματα:
Average waiting time from production to entering the Painting department: 0.016846 seconds
Average production time of Final Product: 0.048397 seconds

Για 1000 εξαρτήματα:
Average waiting time from production to entering the Painting department: 0.016900 seconds
Average production time of Final Product: 0.048677 seconds

Για 2000 εξαρτήματα:
Average waiting time from production to entering the Painting department: 0.016832 seconds
Average production time of Final Product: 0.042239 seconds

Παρατηρούμε δηλαδή πως όσο μεγαλώνουν οι τιμές το αποτέλεσμα σταθεροποιείται και ο λόγος των δύο χρόνων είναι περίπου 1/3.

Για περισσότερες λεπτομέρειες αναφερθείτε στα σχόλια του κώδικα.