Πουρναρας Γεωργιος **Α.Μ. 1115201800162**

Η ιδεα πισω απο την υλοποιηση εχει να κανει με 2 χωρους διαμοιραζόμενης μνημης και 2 σεμαφορους που αφορουν τον εκαστοτε χωρο. Τα προγραμματα P1 , ENC1 διαβαζουν και γραφουν δεδομενα στον πρωτο χωρο ο οποιος εχει key = 1234 , ενω αντισοιχα οι P2 , ENC2 εχουν τον δευτερο χωρο με key=1925. Η CHAN αποτελει τον διαμεσολαβητη και εργαζεται και στους δυο χωρους ενω παραλληλα ειναι μερος και των 2 σεμαφορων.

Memory.c Memory.h

Υλοποιουμε τη κοινη μνημη σαν μια δομη η οποια περιεχει

1. Το μυνημα
2. Τη πιθανοτητα που εδωσε ο χρηστης
3. Τη hash value που υπολιγιζει η ENC? πριν στειλει το μυνημα στην CHAN
4. Εναν counter ο οποιος βοηθα την CHAN να καθοριζει απο ποια διεργασια προερχεται το πιο προσφατο μυνημα
5. Το id του σεμαφορου ο οποιος αφορα τη συγκεκριμενη μνημη
6. Μια flag badnews η οποια χρησιμοποιειται σε περιπτωση αλλαγης του μυνηματος απο την CHAN

Η συναρτηση CopyString αντιγραφει το δωσμενο μυνημα στη κοινη μνημη και κατοπιν προσθετει την τιμη 2 στον ChangeCouner

Η makeSpace δεχεται ενα κλειδι και δημιουργει εναν χωρο κοινης μνημης και επιστρεφει το share memory id. Αν υπαρχει χωρος με αυτο το κλειδι επιστρεφει παλι το shm_id.

Η hash δημιουργει ενα hash code για ενα μυνημα.Χρησιμοποιειται απο τις ENC

semaph.c semaph.h

Οι semUp & semDown εινια βοηθητικες συναρτησεις για τις εντολες wait και signal .

****

Εδω για παραδειγμα στον δευτερο σεμαφορο γιντεται δινεται η εντολη signal

```
semUp(sop,1);
semop(sema_id2,sop,1);
```

****

P?.c P?.h

Η P1 δημιουργει τον χωρο και τον σεμαφορο που θα εργαζεται πανω σε αυτον και επειτα δεχεται το πρωτο μηνυμα και το γραφει στην μνημη. Μεσα στη λουπα “ξυπναει” την ENC1 και “κοιμαται” αυτη . Οταν τη ξυπνησει η ENC1 στον χωρο θα υπαρχει γραμμενο το μυνημα της P2 , το οποιο θα εκτυπωσει και θα ζητησει το επομενο. Ολα αυτα μεχρι να δωθει το μηνυμα TERM οπου θα απελευθερωσει τον δεσμευμενο χωρο και θα τερματισει.

Η P2 , κανει σχεδον τα ιδια. Η μονη διαφορα ειναι οτι κανει access στον ηδη υπαρχοντα χωρο και σεμαφορο και δεν δημιουργει αυτη.

ENC?.c ENC?.h

Η ENC1 κανει access στον ηδη υπαρχοντα χωρο και σεμαφορο και για οσο το μυνημα δεν ειναι TERM

- Κραταει ενα backup του μυνηματος και το στελνει ενα στην CHAN. Αν η CHAN το πειραξει , η ENC2 θα ζητησει νεα μεταδωση. Αυτο επιτυγχανεται με τη μεταβλητη badNews.
- Κανει την δουλεια που εκανε η ENC2 στο δικο της μηνυμα προηγουμενος. Τσεκαρει ,δηλαδη, αν η hash value ειναι ιδια με την πραγματικη .
- Ξυπνα την P1 και κοιμιζει τον εαυτο της

Η ENC2 κανει αρκετα παρομοια πραγματα

CHAN.c CHAN.h

Η CHAN κανει access στον ηδη υπαρχοντα χωρο και σεμαφορο και επειτα δημιουργει και εναν δευτερο για τις P2, ENC2. Ο χωρος που δημιουργει η CHAN εχει counter=0. Αρα οι στην συνθηκη

```
if ((mem1->ChangeCounter)>(mem2->ChangeCounter)){
```

οι τιμες θα ειναι : 1>0..1<2..3>2… κλπ

Επειτα, αν το μυνημα εχει παει στην ENC2 και νωριτερα υπηρξε αλοιωση τοτε εδω ξανακαλειται η ENC1. Επειτα ξαναγινεται randomize στο μυνημα εκτος

αν ειναι το TERM. Γραφεται στην δευτερη μνημη , ενω αντιγραφεται και η hashValue . Ξυπναει η ENC2 και κοιμαται η CHAN. Στο else γινεται μια παρομοια διαδικασια. Τελος ελευθερωνεται ο χωρος που δεσμευθηκε.

Η messageRandomize , αλλαζει τους χαρακτηρες του μυνηματος αν η τιμη της rand ειναι μεγαλυτερη απο την τιμη posibol που εδωσε ο χρηστης.

Σειρα εκτελεσης:

1. make
2. ανοιγμα 5 τερμιναλ στον ιδιο φακελο
3. P1 possibility (0-100) στο πρωτο τερμιναλ // Possibility of message remaining intact
4. γραψε μυνημα στην P1
5. ENC1 στο δευτερο τερμιναλ
6. CHAN στο τριτο τερμιναλ
7. ENC2 στο τεταρτο τερμιναλ
8. P2 στο πεμπτο τερμιναλ (θα εκτυπωθει αμεσως το μυνημα της P1)
9. γραψε μυνημα στην P2
10. (θα γινει εκτυπωση στην P1) γραψε μηνυμα στην P1
11. …..
12. γραψε TERM σε οποιαδηποτε διεργασια για τερματισμο ολων

Τα μηνυματα να ειναι μεχρι 19 χαρακτηρες και να μην εχουν κενα. Αν καποιο απο τα κριτηρια δεν ισχυει το προγραμμα μπορει να χαλασει.