Λειτουργικά Συστήματα

Άσκηση 3

1 Εισαγωγικά

Όταν δύο ή περισσότερες διεργασίες ή νήματα προσπελαύνουν τον ίδιο πόρο (resource) ή γράφουν στις ίδιες θέσεις μνήμης, απαιτείται συγχρονισμός για αποφυγή race conditions. Για τον συγχρονισμό διεργασιών (processes) και νημάτων (threads) παρέχονται τα εξής εργαλεία:

Processes: Semaphores (σημαφόροι)

Threads: Mutexes και condition variables

Στην άσκηση αυτή θα εξοικειωθείτε με τη χρήση των παραπάνω εργαλείων σε διαφορετικά σενάρια. Για τη διεξαγωγή της άσκησης κατεβάστε το σχετικό code template από τη διεύθυνση `http://rio.ceid.upatras.gr:8000/~spyros/os/code2017.tar`.

☞ Για να κάνετε compile τρέξτε απλά `make` μέσα στο directory που παράγεται από το tar.

2 Συγχρονισμός διεργασιών

Σε αυτό το τμήμα της άσκησης καλείστε να γράψετε τα προγράμματα `multiproc_1` και `multiproc_2`.

Το `multiproc_1` παίρνει ένα ή περισσότερα ορίσματα. Το πρώτο όρισμα (υποχρεωτικό) είναι ένας ακέραιος αριθμός. Ας τον ονομάσουμε `rep` (από το repetitions). Τα υπόλοιπα ορίσματα είναι αυθαίρετα strings. Όταν καλείται, το `multiproc_1` πρέπει να δημιουργήσει τόσες διεργασίες-παιδιά όσα και τα strings, και κάθε διεργασία να εκτυπώσει το string που της αντιστοιχεί `rep` φορές. Για την εκτύπωση πρέπει αντί για την `printf()` να χρησιμοποιηθεί η `display(char *)`, η οποία παρέχεται στο `util.c`.

Όταν γράψετε τον κώδικα και τον τρέξετε για πρώτη φορά με δύο ή περισσότερες διεργασίες-παιδιά, θα παρατηρήσετε ότι τα output των διαφορετικών διεργασιών τυπώνονται με τους χαρακτήρες τους ανακατεμένους. Στην άσκηση αυτή θέλουμε να χρησιμοποιήσετε μηχανισμούς για να εξασφαλίσετε την αποκλειστική εκτύπωση από μία διεργασία τη φορά, ώστε οι γραμμές των output να βγαίνουν μεν με *τυχαία* σειρά, αλλά κάθε γραμμή να είναι ''καθαρή'', δηλαδή να *μην* είναι ανακατεμένοι οι χαρακτήρες της με χαρακτήρες άλλων γραμμών.

Προσοχή: Δεν μπορείτε να τροποποιήσετε καθόλου τη `display()`, ούτε και να χρησιμοποιήσετε τη `sleep()` ή τη `usleep()`. Επίσης είναι λάθος να τρέχετε πρώτη τη μία διεργασία, μετά την άλλη, κ.ο.κ. Οι διεργασίες πρέπει να τρέχουν πραγματικά παράλληλα.

Για παράδειγμα, μια σωστή εκτέλεση του `multiproc_1` θα μπορούσε να είναι:

```
$ ./multiproc_1  3  University  Patras
[1000,123456789] University
[1000,123456789] University
[1001,987654321] Patras
[1000,123456789] University
[1001,987654321] Patras
[1001,987654321] Patras
```

όπου η μία διεργασία-παιδί (με PID 1000) τυπώνει 3 φορές ''University'' και η δεύτερη διεργασία-παιδί (με PID 1001) τυπώνει 3 φορές ''Patras''. Για διευκόλυνσή σας στην κατανόηση του τι συμβαίνει, η display τυπώνει και το PID της διεργασίας και το TID του thread.

Το πρόγραμμα `multiproc_2` θα πρέπει να κάνει ακριβώς το ίδιο με το προηγούμενο, αλλά επιπλέον η κάθε διεργασία-παιδί πρέπει να καλεί *μία φορά* τη συνάρτηση `init()` που παρέχεται στο `util.c`, *πριν* τις κλήσεις της `display()`. Και το πιο σημαντικό είναι ότι *καμία* διεργασία-παιδί δεν πρέπει να αρχίσει τις κλήσεις της `display()` πριν να έχουν ολοκληρώσει την κλήση της `init()` *όλες* οι διεργασίες!

3 Συγχρονισμός νημάτων

Σε αυτό το τμήμα της άσκησης καλείστε να γράψετε τα προγράμματα `multithread_1.c` και `multithread_2.c`. Αυτά πρέπει να συμπεριφέρονται ακριβώς όπως τα `multiproc_1` και `multiproc_2`, αντίστοιχα, αλλά αντί για διεργασίες-παιδιά να το επιτυγχάνουν με threads.

Για τα νήματα πρέπει να χρησιμοποιήσετε υποχρεωτικά `pthreads`, που είναι το standard API. Αν και τα εργαλεία συγχρονισμού διεργασιών που χρησιμοποιήσατε στα προηγούμενα προγράμματα θα μπορούσαν να χρησιμοποιηθούν κι εδώ, στα προγράμματα αυτά **οφείλετε να χρησιμοποιήσετε υποχρεωτικά τα εργαλεία συγχρονισμού νημάτων που προσφέρει το API της βιβλιοθήκης `pthreads`**.

4 Οδηγίες παράδοσης και αυτόματου feedback

Για την ορθότερη υλοποίηση της άσκησης και τη διόρθωση σφαλμάτων, έχετε στη διάθεσή σας ένα σύστημα αυτόματης διόρθωσης, το οποίο μπορείτε να χρησιμοποιήσετε όσες φορές θέλετε. Με το σύστημα αυτό, μας στέλνετε την υλοποίησή σας, και μέσα σε λίγα λεπτά σας επιστρέφονται συνοπτικά αποτελέσματα του ελέγχου που κάνουμε.

Τα τεστ στα οποία υποβάλλουμε τις υλοποιήσεις σας σε αυτό το στάδιο πιθανόν να είναι λιγότερα (και λίγο απλούστερα) από το σύνολο των τεστ που θα τρέξουμε για την βαθμολόγηση της άσκησης. Δηλαδή, το ότι περνάτε όλα τα τεστ επιτυχώς δεν συνεπάγεται αυτόματα 10 στα 10.

Για να μας στείλετε την άσκηση, ακολουθήστε τα παρακάτω βήματα:

- Στείλτε μας συνημμένα τα αρχεία `multiproc_[12].c`, `multithread_[12].c`, και `report.pdf` στη διεύθυνση `spyros+hw3@ceid.upatras.gr`.
- Τα αρχεία `util.c` και `util.h` *μην* μας τα στείλετε, θα τα προσθέσουμε ούτως ή άλλως μόνοι μας για να βεβαιωθούμε ότι δεν έχουν τροποποιηθεί. Ακόμα κι αν τα στείλετε θα τα αντικαταστήσουμε με τα δικά μας versions.
- Όπως και στις περασμένες ασκήσεις, το subject του email ΠΡΕΠΕΙ να είναι αποκλειστικά και μόνο ο Αριθμός Μητρώου σας (π.χ., 1234), χωρίς έξτρα κενά, χωρίς "AM: 1234", χωρίς "Re:", χωρίς οτιδήποτε άλλο πέρα από τα τέσσερα αριθμητικά ψηφία του ΑΜ σας. Το email πρέπει να σταλεί από τον λογαριασμό σας στο CEID.
- Θα λάβετε άμεσα επιβεβαίωση για το email σας, και σε λίγα λεπτά θα λάβετε και δεύτερο email που θα σας δίνει μια ένδειξη του πόσα τεστ περάσατε. Το περιεχόμενο των τεστ είναι επίτηδες κρυφό, ώστε να διερευνήσετε όλες τις πιθανές αιτίες σφαλμάτων. Μερικά και μόνο μερικά τεστ παρέχουν κάποιο hint για το είδος του λάθους, αν δεν τα περάσετε.

Μπορείτε να στείλετε την υλοποίησή σας όσες φορές θέλετε. Η **τελευταία** υλοποίηση που θα λάβουμε θα είναι και αυτή που θα βαθμολογηθεί. Όλες οι προηγούμενες θα αγνοηθούν. Αν συνεχίσετε να στέλνετε υλοποιήσεις μετά το τέλος της προθεσμίας, για κάθε μέρα καθυστέρησης θα χρεώνεστε βαθμούς όπως έχει εξηγηθεί αναλυτικά στο πρώτο μάθημα (και στις αντίστοιχες διαφάνειες).

ΣΗΜΑΝΤΙΚΟ: Για την παράδοση της τελικής μορφής της άσκησης οφείλετε να συμπεριλάβετε και ένα σύντομο **report.pdf** 1-2 σελίδων, που να περιγράφει τη λογική που ακολουθήσατε και προβλήματα που αντιμετωπίσατε, **υποχρεωτικά σε format PDF**.

Καλή επιτυχία!