Η/Υ 445: Λειτουργικά Συστήματα
Άνοιξη 2002 - Παναγιώτα Φατούρου
1ο Σετ Προγραμματιστικών Ασκήσεων

Ημερομηνία Παράδοσης: Πέμπτη, 21/3/02

Σχόλια: Θα πρέπει να στείλετε με e-mail την άσκησή σας στη διεύθυνση cs445@cs.uoi.gr μέχρι τις 12:59 της Πέμπτης 21/3. Επίσης, θα πρέπει να κρατήσετε την άσκησή σας σε κάποιο directory στην περιοχή σας χωρίς να αλλάξετε την ημερομηνία της μέχρι να γίνει η εξέτασή της. Η ημερομηνία τελευταίας εγγραφής της άσκησής σας θα πρέπει να είναι είτε κάποια προηγούμενη ή η αναγραφόμενη ημερομηνία και ώρα παράδοσης. Μην ετοιμάζετε την άσκησή σας την τελευταία στιγμή. Η ημερομηνία και ώρα παράδοσης προβλέπει πιθανά προβλήματα στο σύστημα και η άσκησή σας θα πρέπει να παραδοθεί μέχρι αυτή την ημερομηνία ανεξάρτητα από τέτοιου είδους προβλήματα. Σας υπενθυμίζω πως ο κώδικάς σας θα πρέπει να είναι γραμμένος σε C, να τρέχει στα μηχανήματα του εργαστηρίου του Τμήματος, να είναι ευανάγνωστος και να έχει σχολιαστεί κατάλληλα.

Ο σκοπός της άσκησης είναι να υλοποιήσετε δύο προβλήματα διαδιεργασιακής επικοινωνίας στο UNIX (σε C) και να εξοικειωθείτε με κάποιους τρόπους υλοποίησης κρίσιμων περιοχών στο UNIX.

1. **Πρόβλημα Συνδαιτημόνων Φιλοσόφων**

   Υλοποιήστε ένα πρόγραμμα C το οποίο θα δημιουργεί 5 διεργασίες συνδαιτημόνων φιλοσόφων (Ενότητα 2.3.1, Tanenbaum). **Η κοινή μνήμη και οι σημαφόροι θα πρέπει να υλοποιηθούν με χρήση αρχείων.**

   Κάθε φιλόσοφος θα είναι μια ξεχωριστή διεργασία (process) στο UNIX (που θα δημιουργηθεί με την κλήση συστήματος fork). Κάθε σημαφόρος θα υλοποιείται με ένα αρχείο (δείτε manual της open(2) για O_CREAT και O_EXCL, εντολή "man 2 open"), ενώ σε αρχεία θα γράφεται και η κατάσταση κάθε φιλοσόφου. Έτσι, το πρόγραμμα σας θα πρέπει να διαχειρίζεται 2 αρχεία για κάθε φιλόσοφο (το αρχείο σημαφόρο του και το αρχείο στο οποίο ο φιλόσοφος εγγράφει την κατάστασή του), καθώς και ένα αρχείο mutex για υλοποίηση αμοιβαίου αποκλεισμού (σύνολο 11 αρχεία).

   Το πρόγραμμά σας, αφού δημιουργήσει τα απαραίτητα αρχεία, θα πρέπει να δημιουργεί τις 5 διεργασίες φιλοσόφους. Οι διεργασίες αυτές θα πρέπει να χρησιμοποιούν μικρές τυχαίες καθυστερήσεις (δείτε sleep) για να τρέχουν ασύγχρονα.

   Οι διεργασίες φιλόσοφοι θα πρέπει να τυπώνουν μηνύματα στο stderr (π.χ., "philosopher 3 eating", "philosopher 0 raised chopstick 1", κλπ.) που δείχνουν σε ποια ακριβώς κατάσταση βρίσκονται οι ίδιες και τα πηρούνια ανά πάσα στιγμή. Φροντίστε να έχετε αρκετά μηνύματα ώστε να ξέρετε σε ποια ακριβώς κατάσταση βρίσκονται οι διεργασίες και οι πόροι, αλλά όχι περιττά μηνύματα που δεν δίνουν χρήσιμες πληροφορίες (και κουράζουν).

   Οι κλήσεις συστήματος που θα χρησιμοποιήσετε είναι fork, wait, sleep, creat, open, close, unlink, write, read.

   Υλοποιήστε διαδικασίες down και up που θα παίρνουν ως παράμετρο το όνομα του αρχείο-σημαφόρου πάνω στο οποίο η λειτουργία θα πρέπει να εκτελεστεί, και θα επιτελούν τη λειτουργία (αυτό θα σας βοηθήσει να αποφύγετε επανάληψη του ίδιου κώδικα πολλές φορές). Επίσης, χρησιμοποιείστε πίνακες N στοιχείων για να

αποθηκεύσετε τα ονόματα των αρχείων, τους οποίους φροντίστε να δεικτοδοτείτε με το id κάθε διεργασίας φιλοσόφου.

2. Πρόβλημα Κοιμώμενου Κουρέα

Υλοποιήστε σε C το πρόβλημα του Κοιμώμενου Κουρέα (Ενότητα 2.3.3, Tanenbaum). Οι κοινές μεταβλητές θα πρέπει να υλοποιηθούν με κοινή μνήμη, ενώ οι σημαφόροι με τις κλήσεις συστήματος σημαφόρων που παρέχει το UNIX. Πιο συγκεκριμένα, για τη δημιουργία και διαχείριση σημαφόρων θα πρέπει να χρησιμοποοηθούν οι κλήσεις συστήματος semget, semop και semctl. Ομοίως για τη δήλωση και τη διαχείριση κοινών μεταβλητών θα πρέπει να χρησιμοποιηθούν οι κλήσεις συστήματος shmget, shmat, shmctl.

Μια διεργασία γονέας δημιουργεί μια διεργασία κουρέα και N διεργασίες πελάτες (όπου το N θα πρέπει να μπορεί να το καθορίσει ο χρήστης). Το κουρείο θα πρέπει να διαθέτει CHAIRS θέσεις αναμονής (όπου το CHAIRS καθορίζεται επίσης από τον χρήστη).

Όλες οι διεργασίες θα πρέπει να τερματίζουν. Οι διεργασίες πελάτη κάνουν μια παθητική δουλειά (κουρεύονται) που όταν τελειώσει πρέπει να αποχωρούν από το σύστημα. Ο κουρέας θα πρέπει να μετράει τους πελάτες που έχουν κουρευτεί, καθώς επίσης θα πρέπει να γνωρίζει τον αριθμό των πελάτων που έχουν αποχωρήσει επειδή δεν υπάρχει διαθέσιμη θέση. Όταν το άθροισμα αυτό γίνει N θα πρέπει και αυτός να επιστρέφει και να τερματίζει. Προσέξτε ότι για να το πετύχετε αυτό χρειάζεται να χρησιμοποιήσετε μια ακόμη κοινή μεταβλητή (επιπρόσθετα της waiting) που θα μετρά τους πελάτες που αποχωρούν χωρίς να κουρευτούν.

Η διεργασία γονέας θα πρέπει να περιμένει τον τερματισμό όλων των διεργασιών που δημιούργησε και στη συνέχεια αφού ελευθερώσει την μνήμη που έχει δεσμευτεί για τις κοινές μεταβλητές και τους σημαφόρους θα πρέπει επίσης να επιστρέφει (αυτό θα επιφέρει και τον τερματισμό του προγράμματός σας).

Θα πρέπει να ενσωματώσετε στον κώδικα τόσο των πελατών όσο και του κουρέα μια μικρή τυχαία καθυστέρηση (με χρήση της sleep) ώστε να κάνετε το σύστημα ασύγχρονο.

Οι διεργασίες θα πρέπει να τυπώνουν κατάλληλα μηνύματα που θα αποδεικνύουν την ορθότητα του κώδικά σας. Για παράδειγμα, η διεργασία κουρέας θα ήταν καλό να εμφανίζει το μήνυμα: «Κουρεύω, waiting = <τιμή waiting>» κάθε φορά που κουρεύει ένα πελάτη, καθώς επίσης και το μήνυμα «Επιστρέφω, πελάτες που κουρεύτηκαν = <αριθμός πελατών που κουρεύτηκαν>, πελάτες που αποχώρησαν χωρίς να κουρευτούν = <αριθμός πελατών που αποχώρησαν>». Κάθε διεργασία πελάτη θα πρέπει επίσης να τυπώνει είτε «Customer = <αριθμός πελάτη>: Μόλις κουρεύτηκα» ή «Customer = <αριθμός πελάτη>: Δεν βρήκα διαθέσιμη θέση και αποχωρώ», ανάλογα με την κατάστασή της.

Υποδείξεις

Χρησιμοποιείστε την fflush μετά από κάθε εντολή fprintf για να είστε σίγουροι ότι τα μηνύματά σας τυπώνονται έγκαιρα στην οθόνη σας.

Φροντίστε ο κώδικας σας να μην είναι διαφορετικός στη μορφή του από εκείνον που παρουσιάζεται στο βιβλίο σας. Αυτό θα το πετύχετε με χρήση κατάλληλων συναρτήσεων που θα κάνουν κατάλληλες εργασίες και θα αποκρύπτουν λεπτομέρειες υλοποίησης (όπως π.χ., οι down, up που συζητήθηκαν παραπάνω).

Χρησιμοποιείστε τα manuals του UNIX για να βρείτε πως θα υλοποιήσετε δύσκολα σημεία των ασκήσεων. Θα πρέπει να είστε σε θέση να δικαιολογήσετε τυχόν αυθαίρετες αποφάσεις που θα πάρετε, και να περιγράψετε τον τρόπο λύσης που ακολουθήσατε.

```
ERROR: syntaxerror
OFFENDING COMMAND: --%ztokenexec_continue--

STACK:

false
```