ΠΑΝΕΠΙΣΤΗΜΙΟ ΠΑΤΡΩΝ
ΤΜΗΜΑ ΜΗΧΑΝΙΚΩΝ Η/Υ & ΠΛΗΡΟΦΟΡΙΚΗΣ
ΤΟΜΕΑΣ ΛΟΓΙΚΟΥ ΤΩΝ ΥΠΟΛΟΓΙΣΤΩΝ

Λειτουργικά Συστήματα Ι

http://software.hpclab.ceid.upatras.gr/opsys/opsys/index.php

Χειμερινό εξάμηνο

2014-2015

Διεργασίες, νήματα, σημαφόροι, σήματα, κοινή μνήμη, sockets

1 Εισαγωγή

Ο στόχος της άσκησης είναι η εξοικείωση με θεμελιώδεις έννοιες και μηχανισμούς που προσφέρει το λειτουργικό σύστημα. Για το σκοπό αυτό ζητείστε να κάνετε δύο ξεχωριστές υλοποιήσεις. Η μία θα βασίζεται στη δημιουργία και διαχείριση διεργασιών (processes) και την επικοινωνία/συγχρονισμό μεταξύ τους με IPC (interprocess communication), όπως shared memory, pipes, semaphores, signals, κ.τ.λ. Η άλλη θα χτιστεί με τη χρήση νημάτων (threads), και το συγχρονισμό μεταξύ τους για την σωστή διαχείριση της εκ των πραγμάτων κοινής τους μνήμης με mutexes και condition variables. Και οι δύο υλοποιήσεις θα κάνουν χρήση sockets για επικοινωνία client-server.

2 Περιγραφή

2.1 Γενικά

Το θέμα της άσκησης είναι η δημιουργία ενός server για ένα multiplayer online game με συγκεκριμένο αριθμό παικτών. Κάθε νέος παίκτης που συνδέεται στον server επιλέγει στοιχεία για το αρχικό του inventory. Μόλις συμπληρωθεί ο προκαθορισμένος αριθμός παικτών, η παρτίδα ξεκινάει. Νέοι παίκτες που συνδέονται στον server στη συνέχεια αντιστοιχίζονται σε νέα παρτίδα, ενώ πολλές παρτίδες μπορούν να εξελίσσονται ταυτόχρονα. Κατά τη διάρκεια μιας παρτίδας, ο κάθε παίκτης μπορεί να στέλνει μηνύματα τα οποία παραδίδονται μέσω του server σε όλους τους παίκτες της ίδιας παρτίδας. Η παρτίδα τελειώνει μόλις και ο τελευταίος παίκτης (της παρτίδας) αποχωρήσει από το παιχνίδι.

2.2 Αναλυτικά

Πιο συγκεκριμένα, η υλοποίησή σας πρέπει να ικανοποιεί τα εξής:

- Ο server εκτελείται ως εξής:
  `./gameserver –p <num_of_players> -i <game_inventory> -q <quota_per_player>`

όπου η πρώτη παράμετρος είναι ένας integer που δηλώνει τον αριθμό παικτών ανά παρτίδα, και η δεύτερη παρέχει το filename με το συνολικό απόθεμα (inventory) ανά παρτίδα (δες παρακάτω). Η τρίτη παράμετρος (quota) καθορίζει το μέγιστο μέγεθος inventory που αναλογεί σε κάθε παίκτη (δες παρακάτω). Και οι τρεις παράμετροι παραμένουν ίδιες για όλες τις παρτίδες που θα ξεκινήσει αυτό το instance του server.

- Το αρχείο με το inventory περιέχει μία ή περισσότερες γραμμές με το εξής format: <resource_name>\t<quantity>\n. Για παράδειγμα, ένα τέτοιο αρχείο θα μπορούσε να είναι:

```
gold        10
armor       50
ammo        40
lumber      20
magic       15
rock        50
```

- Ένας νέος παίκτης εκτελείται ως εξής:

  `./player –n <name> -i <inventory> <server_host>`

  Η πρώτη παράμετρος δηλώνει το όνομα του παίκτη, η δεύτερη το inventory που επιλέγει για τον εαυτό του ο παίκτης, και η τρίτη το hostname όπου τρέχει ο gameserver.

- Κάθε παίκτης δικαιούται να ζητήσει μέχρι quota αντικείμενα για το δικό του inventory. Για παράδειγμα, αν το quota έχει οριστεί να είναι 4, ένας παίκτης μπορεί να ζητήσει

```
gold        1
armor       1
lumber      1
rock        1
```

  ή εναλλακτικά:

```
ammo        1
magic       2
rock        1
```

  ή ακόμη:

```
magic       4
```

  ή, τέλος, λιγότερα από τα αντικείμενα που δικαιούται:

```
rock        1
```

- Η επικοινωνία μεταξύ παίκτη και server γίνεται TCP, και το πρωτόκολλο επικοινωνίας είναι σε ASCII. Συγκεκριμένα, ο παίκτης ανοίγει μια σύνδεση TCP με τον server, και στέλνει μερικές γραμμές με ASCII δεδομένα. Η πρώτη γραμμή περιέχει απλά το όνομά του, και οι επόμενες τα αντικείμενα που επιλέγει, στο ίδιο format με το inventory file. Κατόπιν, στέλνει μια κενή γραμμή, κι αυτό σηματοδοτεί το τέλος της αίτησης συμμετοχής του. Ο server απαντά με “OK” ή με σχετικό μήνυμα λάθους, ακολουθούμενο από ’\n’.
- Τα αντικείμενα στο συνολικό inventory είναι **αναλώσιμα** και όχι ανεξάντλητα! Για να δοθούν κάποια αντικείμενα σε έναν παίκτη, **πρέπει να υπάρχουν διαθέσιμα** στο

συνολικό inventory της παρτίδας. Για παράδειγμα, στο παραπάνω παράδειγμα που ο server έχει “gold 10”, αν δύο παίκτες προλάβουν και πάρουν από τέσσερα, ένας τρίτος παίκτης δεν μπορεί να πάρει παρά το πολύ δύο.

- Αν κάποιο από τα αντικείμενα που ζητάει ο παίκτης είναι άγνωστο στον server, ή αν έχει ήδη εξαντληθεί λόγω επιλογών προηγούμενων παικτών, ή αν ο παίκτης ζητάει περισσότερα αντικείμενα από το quota του, ο server θα τού επιστρέψει ένα μήνυμα λάθους και θα διακόψει άμεσα την σύνδεση με τον συγκεκριμένο παίκτη. Προφανώς, ο παίκτης αυτός δεν μετράει στον συνολικό αριθμό παικτών που πρέπει να συγκεντρωθούν για να ξεκινήσει η παρτίδα.
- Μέχρι να συγκεντρωθεί ο απαιτούμενος αριθμός παικτών για να ξεκινήσει μία παρτίδα, οι ήδη συνδεδεμένοι παίκτες λαμβάνουν κάθε 5 sec ένα μήνυμα από τον server ότι είναι σε αναμονή.
- Όταν ξεκινάει μια νέα παρτίδα, το συνολικό inventory του server αρχικοποιείται στις αρχικές τιμές (π.χ., gold 10, armor 50, κ.τ.λ.).
- Μηνύματα που στέλνονται από τους παίκτες στον server πριν την έναρξη της παρτίδας, αγνοούνται.
- Μηνύματα που λαμβάνει ο παίκτης, τυπώνονται απευθείας στην οθόνη (stdout). Για το λόγο αυτό, ο server στέλνει στους παίκτες μόνο μηνύματα σε ASCII.
- Κατά τη συμπλήρωση του απαιτούμενου αριθμού παικτών, ο server στέλνει μήνυμα “START” σε όλους τους παίκτες αυτής της παρτίδας.
- Κατά την διάρκεια μιας παρτίδας, ό,τι μήνυμα στέλνεται από κάποιον παίκτη στον server, προωθείται από τον server σε όλους τους υπόλοιπους παίκτες της ίδιας παρτίδας.
- Κάποιος παίκτης αποχωρεί απλά με το να κλείσει την σύνδεσή του με τον server. Η παρτίδα συνεχίζει με τους υπόλοιπους.
- Όταν αποχωρήσει και ο τελευταίος παίκτης, η παρτίδα τερματίζεται.
- Ο gameserver αυτός καθεαυτός τερματίζεται με ένα SIGINT interrupt (δηλαδή με CTRL-C).

3 Λεπτομέρειες υλοποίησης

Η άσκηση αυτή είναι 2-σε-1 ☺ Δηλαδή πρόκειται για δύο ανεξάρτητες υλοποιήσεις της ίδιας λειτουργικότητας, η μία βασισμένη σε διεργασίες (processes) κι η άλλη βασισμένη σε νήματα (threads).

Και στις δύο υλοποιήσεις η επικοινωνία παίκτη-server θα γίνεται με τον ίδιο ακριβώς τρόπο, δηλαδή με μια σύνδεση TCP που εγκαθιστά ο παίκτης (με connect()) με τον server ο οποίος ήδη βρίσκεται σε κατάσταση αναμονής (με accept()).

3.1 Υλοποίηση με βάση διεργασίες

Με βάση διεργασίες, θα κάνετε χρήση της fork() για τη δημιουργία νέων διεργασιών.

- Ο server θα περιμένει με accept() τη σύνδεση νέου παίκτη, και θα δημιουργεί **αμέσως μετά την accept()** μια νέα διεργασία για να αναλάβει την επικοινωνία με αυτόν τον παίκτη. Ο ίδιος ο server αμέσως θα μπαίνει πάλι σε κατάσταση αναμονής, περιμένοντας την επόμενη αίτηση.
- Για να συντονίζονται οι διεργασίες των παικτών σχετικά με τα εναπομείναντα αντικείμενα στο απόθεμα ή σχετικά με την συμπλήρωση του απαιτούμενου αριθμού παικτών για να ξεκινήσει η παρτίδα, θα πρέπει να κάνουν χρήση IPC (inter-process communication) για επικοινωνία και συγχρονισμό μεταξύ τους. Τα εργαλεία IPC είναι η κοινή μνήμη (shared memory), οι σημαφόροι (semaphores), οι σωλήνες (pipes), και τα σήματα (signals). Μελετήστε και αποφασίστε ποιο ή ποια από τα εργαλεία αυτά θα πρέπει να χρησιμοποιήσετε.
- Για τον συγχρονισμό των διεργασιών, το busy waiting απαγορεύεται!
- Βεβαιωθείτε ότι το σύστημα δεν καταλήγει σε deadlock υπό οποιεσδήποτε συνθήκες. Οι χρονισμοί για τις διάφορες λειτουργίες (π.χ., την εγγραφή νέου παίκτη) πρέπει να θεωρηθούν ότι μπορούν να πάρουν αυθαίρετο χρόνο για την λειτουργία τους.
- Οι πόροι θα πρέπει να ελευθερώνονται όταν δεν χρειάζονται πλέον. Για παράδειγμα, οι διεργασίες που εξυπηρετούν παίκτες θα πρέπει να τερματίζονται κατάλληλα, ώστε στο σύστημα να μην παραμένουν άχρηστες διεργασίες ή zombies. Σημαφόροι και κοινή μνήμη θα πρέπει να ελευθερώνονται πλήρως κατά τον τερματισμό μιας παρτίδας, καθώς και του server. Μνήμη που δεν χρησιμοποιείται πλέον πρέπει να ελευθερώνεται, ώστε το σύστημα να μπορεί να τρέχει για απεριόριστο χρόνο και αριθμό παρτίδων, αποφεύγοντας τη σπατάλη μνήμης στο πέρασμα του χρόνου. Γενικότερα, δώστε προσοχή να μην αφήνετε ανοιχτούς πόρους που δεν χρειάζονται πλέον οι διεργασίες.
- Για τη δημιουργία νέων διεργασιών, χρησιμοποιείτε τη fork(), που άλλωστε είναι ο μοναδικός τρόπος δημιουργίας νέων διεργασιών σε Unix συστήματα.
- Για τη χρήση σημαφόρων, χρησιμοποιήστε τις sem_post() και sem_wait().
- Για τον έλεγχο τερματισμού κάποιας διεργασίας, δεσμεύστε το σήμα SIG_CHLD.

3.2 Υλοποίηση με βάση νήματα

Ο server θα είναι μια διεργασία που θα εκτελείται στο παρασκήνιο (background). Ο server θα δεσμεύει ένα τμήμα μνήμης ανά παρτίδα, στο οποίο θα αποθηκεύονται όλες οι απαραίτητες πληροφορίες για την λειτουργία της παρτίδας.

- Για κάθε αίτηση που θα λαμβάνει ο server θα δημιουργεί νέα νήματα, τα οποία θα εξυπηρετούν την αίτηση. Ο server αμέσως θα μπαίνει πάλι σε κατάσταση αναμονής, περιμένοντας την επόμενη αίτηση.
- Αντί για fork(), σε αυτή την άσκηση θα χρησιμοποιήσετε την εντολή δημιουργίας νημάτων pthread_create().
- Αντί για sem_wait() και sem_post() θα χρησιμοποιήσετε τις εντολές pthread_mutex_lock(), pthread_mutex_trylock() και pthread_mutex_unlock() για

βραχυπρόθεσμο κλείδωμα τμήματος κώδικα (με σκοπό τον αμοιβαίο αποκλεισμό νημάτων κατά την πρόσβαση σε ένα κρίσιμο τμήμα του κώδικα).

- Αντί για sem_wait() και sem_post() για πιο **μακροπρόθεσμο** κλείδωμα και συγχρονισμό μεταξύ νημάτων θα χρησιμοποιήσετε τις λειτουργίες των condition variables pthread_cond_init(), pthread_cond_wait(), pthread_cond_signal().
- Για να συγχρονίσετε τα νήματα έτσι ώστε, π.χ., ένα νήμα να περιμένει τον τερματισμό ενός άλλου, χρησιμοποιήστε την συνάρτηση pthread_join().
- Πρέπει να δώσετε προσοχή στο γεγονός ότι τα νήματα έχουν όλα πρόσβαση στην ίδια μνήμη της διεργασίας που τα δημιούργησε.
- Τέλος, τα νήματα που εξυπηρετούν τις αιτήσεις θα πρέπει να τερματίζονται κατάλληλα. Γενικότερα, δώστε προσοχή να μην αφήνετε ανοιχτούς πόρους που δεν χρειάζονται πλέον.

4 Βοηθητικοί σύνδεσμοι

Για την ανάπτυξη της άσκησης, μπορείτε να συμβουλευθείτε τις παρουσιάσεις των φροντιστηρίων που είναι αναρτημένες στο site του μαθήματος:
http://software.hpclab.ceid.upatras.gr/opsys/opsys/index.htm
καθώς και τους παρακάτω συνδέσμους:

Εγχειρίδια (3.1)
http://software.hpclab.ceid.upatras.gr/opsys/opsys/project/Socket.htm
http://software.hpclab.ceid.upatras.gr/opsys/opsys/project/server_client.htm
http://software.hpclab.ceid.upatras.gr/opsys/opsys/project/Signals.htm
http://software.hpclab.ceid.upatras.gr/opsys/opsys/project/SharedMemory.htm
http://www.gnu.org/software/make/manual/make.html#Simple-Makefile
http://man7.org/linux/man-pages/man7/sem_overview.7.html
http://man7.org/linux/man-pages/man2/fork.2.html
http://man7.org/linux/man-pages/man7/signal.7.html

Δείγματα κώδικα (3.1)
http://software.hpclab.ceid.upatras.gr/opsys/opsys/project/code/testSocket-1.c
http://software.hpclab.ceid.upatras.gr/opsys/opsys/project/code/testSocket-2.c
http://software.hpclab.ceid.upatras.gr/opsys/opsys/project/code/testSignal-1.c
http://software.hpclab.ceid.upatras.gr/opsys/opsys/project/code/testSignal-2.c
http://software.hpclab.ceid.upatras.gr/opsys/opsys/project/code/testSignal-3.c
http://software.hpclab.ceid.upatras.gr/opsys/opsys/project/code/testSignal-4.c

Εγχειρίδια (3.2)
http://software.hpclab.ceid.upatras.gr/opsys/opsys/project/Socket.htm
http://software.hpclab.ceid.upatras.gr/opsys/opsys/project/server_client.htm
http://software.hpclab.ceid.upatras.gr/opsys/opsys/project/Signals.htm
http://software.hpclab.ceid.upatras.gr/opsys/opsys/project/POSIXThreads.htm
http://software.hpclab.ceid.upatras.gr/opsys/opsys//project/pthread.htm

http://www.gnu.org/software/make/manual/make.html#Simple-Makefile
http://man7.org/linux/man-pages/man7/pthreads.7.html
http://man7.org/linux/man-pages/man7/signal.7.html

Δείγματα κώδικα (3.2)
http://software.hpclab.ceid.upatras.gr/opsys/opsys/project/threads/thread_condition.c
http://software.hpclab.ceid.upatras.gr/opsys/opsys/project/threads/thread_create.c
http://software.hpclab.ceid.upatras.gr/opsys/opsys/project/threads/thread_join.c
http://software.hpclab.ceid.upatras.gr/opsys/opsys/project/threads/thread_mutex.c
http://software.hpclab.ceid.upatras.gr/opsys/opsys/project/threads/thread_timed_wait.c

5 Διαδικαστικά

- Η άσκηση θα υλοποιηθεί είτε ατομικά, είτε σε ομάδες των 2 ατόμων.
- Η ημερομηνία παράδοσης της άσκησης είναι η **Δευτέρα 19 Ιανουαρίου 2015 και ώρα 23:59**.
- Ο κώδικας που θα παραδώσετε θα πρέπει να είναι καλά δομημένος.
- Ο κώδικας που θα παραδώσετε θα πρέπει να είναι καλά σχολιασμένος.
- Ο κώδικας θα πρέπει να χρησιμοποιεί τα εργαλεία που αναφέρθηκαν νωρίτερα (fork, exit, sockets, shared memory, signals, semaphores).
- Ο κώδικας θα αναπτυχθεί σε Linux, και θα ελεγχθεί συγκεκριμένα σε Ubuntu.
- Ο έλεγχος του κώδικα θα γίνει σε 2 φάσεις:
    - Στην πρώτη φάση θα εκτελεστεί η εντολή **`make`** στον κατάλογο που υπάρχει το πρόγραμμα. Εάν το πρόγραμμα μεταγλωττιστεί κανονικά, περνάμε στην επόμενη φάση.
    - Η επόμενη φάση ελέγχει την σωστή λειτουργία του κώδικα.
- Θα φροντίσετε να υπάρχει ένα directory με όνομα **testing**, στο οποίο θα περιέχεται εκτελέσιμο shell script με το όνομα **run_tests**. Η εκτέλεση του script αυτού θα τρέχει κάποιο παράδειγμα. Δηλαδή, θα σηκώνει έναν gameserver και θα τρέχει μερικούς players με προκαθορισμένα (από εσάς) αρχεία inventories που θα βρίσκονται ήδη μέσα στο testing.
- Μαζί με τον κώδικα, θα δώσετε και μια **σύντομη αναφορά** όπου θα περιγράφετε την μεθοδολογία που ακολουθήσατε (όχι κώδικας εδώ). Θα περιγράφετε τις δομές/σήματα που χρησιμοποιήσατε και τις συναρτήσεις/αρχεία με τα οποία αλληλεπιδράτε με αυτές. Η αναφορά θα αποθηκευτεί στο αρχείο **report.pdf**.
- Σε περίπτωση καθυστέρησης στην παράδοση της άσκησης θα υπάρχει μείωση 10% για κάθε 24 ώρες παράδοσης.
- Σε περίπτωση που εντοπιστεί αντιγραφή, ο βαθμός θα μηδενίζεται για όλους όσους εμπλέκονται (και αυτούς που έλαβαν την άσκηση και αυτούς που την έδωσαν).

6 Τρόπος Αποστολής

Η αποστολή της άσκησης θα γίνει μέσω ηλεκτρονικού ταχυδρομείου στις διευθύνσεις spyros--os@cs.vu.nl (δύο παύλες) ΚΑΙ makri@ceid.upatras.gr. Για την ακρίβεια θα στείλετε ένα mail για την υλοποίηση με processes, κι ένα για την υλοποίηση με threads.

Η διεύθυνση από την οποία θα στείλετε την άσκηση, θα πρέπει να είναι της μορφής user@ceid.upatras.gr και όχι προσωπικό e-mail τύπου gmail ή yahoo.

Η ώρα παράδοσης της άσκησης θεωρείται η ώρα αποστολής του κάθε email.

Το κάθε email θα είναι δομημένο ως εξής:

- **Θέμα:** Το θέμα των emails θα είναι της μορφής **AM1_AM2_processes** και **AM1_AM2_threads** όπου AM1 και AM2 οι αριθμοί μητρώου των συνεργατών. Σε περίπτωση που η εργασία γίνει μόνο από ένα άτομο, τότε θα έχουν την μορφή **AM1_processes** και **AM1_threads**.
- **Κυρίως κείμενο (body):** Κενό.
- **Επισυναπτόμενο αρχείο (attachment):** Το email θα περιέχει **ένα** μόνο attachment σε μορφή .tar, το οποίο θα περιέχει τα αρχεία της άσκησης. Το όνομα του attachment θα είναι το ίδιο με το θέμα του email, δηλαδή θα έχει την μορφή **AM1_AM2_processes.tar** ή **AM1_AM2_threads.tar**.
- **Περιεχόμενα επισυναπτόμενου αρχείου .tar:** Το αρχείο .tar θα περιέχει τα αρχεία του πηγαίου κώδικα σε C, το αρχείο **Makefile**, το directory με το παράδειγμα εκτέλεσης **testing** και ένα αρχείο με την αναφορά (σε μορφή pdf) με τον τίτλο **report.pdf.** Τα παραπάνω αρχεία καθώς και το testing/ δεν θα περιέχονται σε κανέναν υποκατάλογο του αρχείου tar.