Κ24: Προγραμματισμός Συστήματος
2η Εργασία – Εαρινό Εξάμηνο '16
Προθεσμία Υποβολής: Κυριακή 24 Απριλίου 2016 Ώρα 23:59

Εισαγωγή στην Εργασία:

Ο στόχος της εργασίας αυτής είναι να εξοικειωθείτε με την δημιουργία διεργασιών (processes) με χρήση των κλήσεων συστήματος fork/exec, την επικοινωνία διεργασιών μέσω named–pipes, τη χρήση low level I/O, το χειρισμό signals και τη δημιουργία bash scripts.

Καλείστε να υλοποιήσετε ένα πρόγραμμα που φτιάχνει «πίνακες ανακοινώσεων» (bulletin boards) με σκοπό την (μονόδρομη) επικοινωνία μεταξύ χρηστών και του δημιουργού ενός πίνακα σε ένα μηχάνημα. Κάθε χρήστης μπορεί να γίνει δημιουργός ενός ή και περισσοτέρων πινάκων ανακοινώσεων. Επίσης κάθε πίνακας μπορεί να οργανωθεί σε κανάλια (channels), στα οποία χρήστες –που μπορεί να είναι διαφορετικοί από τον δημιουργό– θα έχουν τη δυνατότητα να αφήσουν μηνύματα και αρχεία. Τα εν λόγω μηνύματα και αρχεία μπορεί να «δει» μόνο ο δημιουργός του πίνακα. Το board θα πρέπει να παρέχει τη δυνατότητα εμφάνισης των μηνυμάτων και αρχείων που βρίσκονται σε κάθε κανάλι. Τέλος θα πρέπει να δημιουργήσετε ένα bash script για την εξαγωγή στατιστικών στοιχείων για τους ενεργούς και ανενεργούς πίνακες.

Η υλοποίηση σας θα πρέπει να αποτελείται από δύο βασικά προγράμματα, το `board` και το `boardpost`. Το πρώτο εκτελείται από τον χρήστη ο οποίος δημιουργεί τον πίνακα ανακοινώσεων, ενώ το δεύτερο εκτελείται από τους χρήστες που επιθυμούν να στείλουν στον δημιουργό του board μηνύματα/αρχεία σε κάποιο κανάλι του πίνακα. Η επικοινωνία όλων των διεργασιών επιτυγχάνεται με τη χρήση named-pipes.

Το πρόγραμμα board αναλαμβάνει:

- Τη δημιουργία ενός πίνακα ή τη σύνδεση με ήδη υπάρχοντα.
- Τη δημιουργία/διαχείριση των καναλιών καθώς και την (προσωρινή) αποθήκευση των μηνυμάτων/αρχείων ανά κανάλι.
- Την εμφάνιση των μηνυμάτων / αποθήκευση αρχείων που βρίσκονται σε κάθε κανάλι.
- Την έξοδο από το πρόγραμμα board (το server process θα συνεχίσει να εκτελείται).
- Τον τερματισμό του πίνακα και της διεργασίας του server, όπως επίσης και την διαγραφή των αρχείων του board (named-pipes).

Το πρόγραμμα boardpost αναλαμβάνει:

- Την εμφάνιση των διαθέσιμων καναλιών.
- Τη δημιουργία ενός νέου μηνύματος σε κάποιο από τα διαθέσιμα κανάλια.
- Τη μεταφόρτωση (uploading) ενός αρχείου σε κάποιο από τα διαθέσιμα κανάλια.

Περιγραφή του board:

Το πρόγραμμα εκτελείται από τη γραμμή εντολών ως εξής:
% ./board <path>

Το <path> αποτελεί έναν μονοπάτι στο δίσκο, κοινό μεταξύ των προγραμμάτων, για τη δημιουργία του `named-pipe`. Για δοκιμές μπορείτε να χρησιμοποιήσετε τον κατάλογο `/tmp/` στα μηχανήματα της σχολής και να φτιάξετε δικό σας υποκατάλογο κάτω από αυτόν (π.χ., `/tmp/sdi1100666/`). Κάτω από τον υποκατάλογο αυτό, μπορείτε να επινοήσετε pathnames για κάθε board που φτιάχνετε (π.χ., `/tmp/sdi1100666/myboard1`) και να αλλάξετε την προσβασιμότητα στα board αλλάζοντας τα δικαιώματα των αντίστοιχων αρχείων στο δίσκο.

Όταν τρέξει το ./`board` δέχεται από το standard input εντολές της εξής μορφής:

1. `createchannel <id> <name>`
   Η εντολή αυτή δημιουργεί ένα κανάλι σε ένα πίνακα. Το <id> αποτελεί το αναγνωριστικό του καναλιού και είναι μοναδικό. Το <name> αποτελεί το όνομα του καναλιού.
2. `getmessages <id>`
   Η εντολή αυτή εμφανίζει όλα τα μηνύματα και αρχεία τα οποία υπάρχουν στο κανάλι με αναγνωριστικό <id>. Στην περίπτωση αρχείων, αυτά πρέπει να σώζονται στο δίσκο με το αντίστοιχο όνομα. Αν το όνομα του αρχείου ήδη υπάρχει, θα πρέπει να φροντίσετε να προστίθεται κατάλληλο επίθεμα.
3. `exit`
   Η εντολή αυτή τερματίζει το board client, ενώ το server process του board συνεχίζει να εκτελείται.
4. `shutdown`
   Η εντολή αυτή τερματίζει το board client, το board server, και διαγράφει τα αρχεία του board (για παράδειγμα τα named-pipes.

Περιγραφή του `boardpost`:

Το `boardpost` αποτελεί το εκτελέσιμο το οποίο πρέπει να έχει ένα χρήστης για να είναι σε θέση να επικοινωνήσει με ένα board. Η επίκληση γίνεται ως εξής:
`% ./boardpost <path>`

Το <path> είναι αυτό που χρησιμοποιήθηκε κατά τη δημιουργία του board (π.χ. `/tmp/sdi1100666/myboard1`). Όταν τρέξει το `./boardpost`, δέχεται από το standard input εντολές της εξής μορφής:

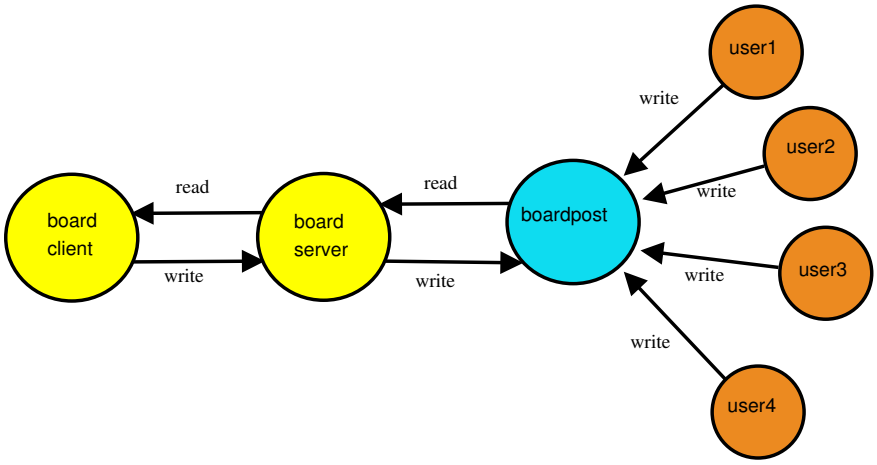
1. `list`
   Η εντολή αυτή εμφανίζει όλα τα διαθέσιμα κανάλια του Πίνακα Ανακοινώσεων, με το όνομα και το μοναδικό `id` του κάθε καναλιού.
2. `write <id> <message>`
   Η εντολή αυτή αποθηκεύει το μήνυμα <message> στο κανάλι με αναγνωριστικό <id>.
3. `send <id> <file>`
   Η εντολή αυτή αποστέλλει το αρχείο <file> στο κανάλι με αναγνωριστικό το <id>.

Η Υλοποίηση του `board`:

Την πρώτη φορά που θα εκτελεστεί το πρόγραμμα `./board` για ένα συγκεκριμένο board pathname θα πρέπει να δημιουργεί ένα "server" process που εξυπηρετεί τον πινάκα ανακοινώσεων. Δηλαδή το πρόγραμμα board παράγει *δύο διεργασίες*: μία client (τη διεργασία που ξεκινάει όταν τρέξουμε το πρόγραμμα) και μία server (που γίνεται fork από την προηγούμενη). Αυτό το server process θα συνεχίσει να τρέχει και αφού η διεργασία του `./board` που διαβάζει εντολές από το standard input έχει τερματίσει. Όλες οι άλλες διεργασίες (τόσο από την τωρινή ή μελλοντικές εκτελέσεις του προγράμματος `./board` όσο και του `./boardpost`) επικοινωνούν με αυτή την server process για να υλοποιήσουν εντολές που αφορούν στο συγκεκριμένο board. Το Σχήμα 1 παρουσιάζει μια πιθανή λειτουργική αλληλεπίδραση των εμπλεκομένων διεργασιών.

Κατά την εκτέλεση του `./board` θα πρέπει να δημιουργηθούν 2 named-pipes (ή 4, αν θέλετε να έχετε ευχέρεια να διαβάζετε και να γράφετε σε ξεχωριστά κανάλια όπως δείχνει και το Σχήμα 1). Προτείνουμε στο δίσκο να έχουν τα ονόματα <path>`_self` και <path>`_others`, για τον πίνακα που αντιστοιχεί στο <path>.

Το <path>`_self` είναι το κανάλι επικοινωνίας μεταξύ της διεργασίας-client του προγράμματος `./board` και της διεργασίας-server του `board`. Μέσω του καναλιού αυτού, ο server παίρνει εντολές για να παραδώσει μηνύματα στον παραλήπτη (π.χ., '`getmessages`') και επιστρέφει τα αντίστοιχα δεδομένα. Το ακριβές πρωτόκολλο επικοινωνίας πάνω από το κανάλι (π.χ., αν η μία διεργασία όταν λάβει `getmessages` από το χρήστη γράφει στο κανάλι



Σχήμα 1: Αλληλεπίδραση διεργασιών για την υλοποίηση των «Πινάκων Ανακοινώσεων»

τη λέξη `GET`) είναι δική σας σχεδιαστική επιλογή.

Το <path>_others είναι το κανάλι επικοινωνίας μεταξύ του προγράμματος ./boardpost και της διεργασίας-server του board. Μέσω του καναλιού αυτού, ο server παίρνει εντολές για να αποθηκεύσει μηνύματα (π.χ., σαν αποτέλεσμα εντολών '`write`' ή '`send`' από το χρήστη) ώστε να διαβαστούν αργότερα από τον board client.

Όταν ξεκινάει το πρόγραμμα ./board θα πρέπει να ελέγξει αν το server process του πίνακα ανακοινώσεων ήδη υπάρχει και τι process id έχει. Μπορείτε να χρησιμοποιήσετε ένα αρχείο <path>_pid που να κρατάει το process id του αντίστοιχου server process. Αυτό δεν είναι απαραίτητο (εξαρτάται από άλλες σχεδιαστικές αποφάσεις) αλλά θα σας φανεί χρήσιμο σε περίπτωση που χρειαστείτε να στείλετε σήμα (`signal`) στο server process, όπως συζητάμε παρακάτω.

Η Υλοποίηση του boardpost:

Ένας χρήστης μπορεί να αλληλεπιδράσει με ένα πίνακα καλώντας το πρόγραμμα ./boardpost. Ο χρήστης που τρέχει το ./boardpost πιθανόν δεν είναι ο ίδιος που δημιούργησε το board αλλά άλλος χρήστης του συστήματος που του έχουν δοθεί το εκτελέσιμο και δικαιώματα στα αντίστοιχα path. Το πρόγραμμα επικοινωνεί με τον server του board, όπως αναφέρεται παραπάνω. Για μεταφορά αρχείων θα πρέπει να χρησιμοποιήσετε low-level I/O αφού τα εν λόγω αρχεία μπορεί να είναι όχι μόνο αρχεία κειμένου αλλά και δυαδικά.

Bash Scripting:

Για να βλέπετε γρήγορα ποιοι πίνακες είναι ενεργοί, ποια server processes μπορεί να έχουν πεθάνει, κτλ., θα πρέπει να αναπτύξετε το `boardstatus.sh` script το οποίο θα εκτελείται ως εξής:

```
./boardstatus.sh <path>
```

Το <path> αποτελεί έναν κατάλογο που περιλαμβάνει πολλαπλά boards. Για τα μηχανήματα του τμήματος αυτός ο κατάλογος θα μπορούσε να είναι ο /tmp/<username>. Το `boardstatus.sh` θα πρέπει να μπορεί να αναγνωρίσει πόσα από τα boards είναι ενεργά (δηλαδή έχουν και named-pipe στο δίσκο και server process που τρέχει) και πόσα ανενεργά, καθώς επίσης και να μπορεί να τυπώσει τα ονόματα αυτών. Για παράδειγμα:

```
./boardstatus.sh <path>
2 Boards Active:
<path>/general
```

```
<path>/playground
1 inactive Board (on disk but no server running):
<path>/myboard1
```

Για τον έλεγχο της κατάστασης (status) κάθε πίνακα, θα πρέπει είτε να κρατάτε (στο προαναφερθέν αρχείο `<path>_pid`) είτε να είστε σε θέση να ανακτήσετε (με κατάλληλη εντολή `ps`), το pid του server process για κάθε πίνακα που εξυπηρετείτε.

Άλλες Σημαντικές Απαιτήσεις:

- Η server process ενός πίνακα επικοινωνεί ταυτόχρονα σε δύο κανάλια που είναι και τα δύο αμφίδρομα. Προφανώς, δεν πρέπει η υλοποίησή σας να μπλοκάρει για πάντα στο ένα κανάλι αφού είναι πολύ πιθανόν να χρειάζεται ταυτόχρονα προσοχή και το δεύτερο κανάλι. Μπορείτε να χρησιμοποιήσετε σήματα (`signals`) για ξεμπλοκάρισμα, ή να ανοίξετε το named pipe χωρίς να μπλοκάρει σε `read` (ή και να χρησιμοποιήσετε την κλήση συστήματος `select`).
- Το *Cntrl–C* (σήμα *SIGINT*) θα πρέπει να αγνοείται και στο ./`board` και στο ./`boardpost`.

Διαδικαστικά:

- Το πρόγραμμά σας θα πρέπει να τρέχει στα μηχανήματα Linux της σχολής.
- Για επιπρόσθετες ανακοινώσεις, παρακολουθείτε το forum του μαθήματος στο `piazza.com` για να δείτε ερωτήσεις/απαντήσεις/διευκρινήσεις που δίνονται σχετικά με την άσκηση. Η παρακολούθηση του φόρουμ στο `piazza.com` είναι *υποχρεωτική*.

Τι πρέπει να Παραδοθεί:

1. Όλη η δουλειά σας σε ένα tar-file που να περιέχει όλα τα `source files, header files` και `Makefile`. Προσοχή: φροντίστε να τροποποιήσετε τα δικαιώματα αυτού του αρχείου πριν την υποβολή, π.χ. με την εντολή:
   `% chmod 755 OnomaEponymoProject2.tar`
   Για πιο πολλές λεπτομέρειες δείτε τη σελίδα του μαθήματος.
2. Ένα `README` (απλό text αρχείο) με κάποια παραδείγματα μεταγλώττισης και εκτέλεσης του προγράμματός σας. Επίσης, μπορείτε να συμπεριλάβετε άλλες διευκρινίσεις που κρίνετε απαραίτητες (για τυχόν παραδοχές που έχετε κάνει, κτλ.).
3. Οποιαδήποτε πηγή πληροφορίας, συμπεριλαμβανομένου και κώδικα που μπορεί να βρήκατε στο Διαδίκτυο θα πρέπει να αναφερθεί και στον πηγαίο κώδικά σας αλλά και στο παραπάνω `README`.

Τι θα Βαθμολογηθεί:

1. Η συμμόρφωση του κώδικά σας με τις προδιαγραφές της άσκησης.
2. Η οργάνωση και η αναγνωσιμότητα (μαζί με την ύπαρξη σχολίων) του κώδικα.
3. Η χρήση `Makefile` και η κομματιαστή σύμβολο-μετάφραση (separate compilation).

Άλλες Σημαντικές Παρατηρήσεις:

1. Οι εργασίες είναι *ατομικές*.
2. Όποιος υποβάλλει/δείχνει κώδικα που δεν έχει γραφτεί από την ίδια/τον ίδιο *μηδενίζεται* στο μάθημα.
3. Αν και αναμένεται να συζητήσετε με φίλους και συνεργάτες το πως θα επιχειρήσετε να δώσετε λύση στο πρόβλημα, *αντιγραφή κώδικα* (οποιασδήποτε μορφής) είναι κάτι που *δεν επιτρέπεται* και δεν πρέπει να γίνει. Οποιοσδήποτε βρεθεί αναμεμειγμένος σε αντιγραφή κώδικα *απλά παίρνει μηδέν* στο μάθημα. Αυτό ισχύει για *όλους όσους εμπλέκονται*, ανεξάρτητα από το ποιος έδωσε/πήρε κλπ.
4. Το πρόγραμμά σας θα πρέπει να γραφτεί σε C (ή C++ χωρίς όμως STL extensions).