Σκοπός

Ο στόχος αυτής της εργασίας είναι να εξοικειωθείτε με τον προγραμματισμό συστήματος σε Unix περιβάλλον και συγκεκριμένα, με τη δημιουργία νημάτων και τη δικτυακή επικοινωνία. Η εργασία χωρίζεται σε τρία μέρη.

Problem 1: Multi-threaded network server (50 points).
Περιγραφή - Εκτέλεση.
Γράψτε έναν πολυνηματικό εξυπηρετητή δικτύου σε C/C++ που ονομάζεται poller το οποίο εκτελεί μια προεκλογική δημοσκόπηση κομμάτων. Το πρόγραμμα θα το τρέχετε από τη γραμμή εντολής με τα ακόλουθα ορίσματα

```
prompt> poller [portnum] [numWorkerthreads] [bufferSize] [poll-log][poll-stats]
```

Συγκεκριμένα τα ορίσματα είναι:

- **portnum:** ο αριθμός θύρας που ακούει ο εξυπηρετητής
- **numWorkerthreads:** ο αριθμός νημάτων εργατών που θα δημιουργήσει για να κάνει τη δημοσκόπηση. Πρέπει να είναι > 0.
- **bufferSize:** το μέγεθος ενός buffer που θα κρατά συνδέσεις από πελάτες που περιμένουν να εξυπηρετηθούν. Πρέπει να είναι > 0.
- **poll-log:** το όνομα ενός αρχείου όπου θα αποθηκεύονται τα ονόματα χρηστών και οι ψήφοι τους.
- **poll-stats:** το όνομα ενός αρχείου όπου θα αποθηκεύονται τα αποτελέσματα (π.χ., ο αριθμός χρηστών που έχουν ψηφίσει καθε κόμμα).

Τα ορίσματα είναι όλα υποχρεωτικά και πρέπει να δίνονται με τη συγκεκριμένη σειρά και μόνο.

Για παράδειγμα, αν τρέξετε το πρόγραμμα σας με τα ακόλουθα ορίσματα:

```
prompt> poller 5634 8 16 pollLog.txt pollStats.txt
```

τότε ο εξυπηρετητής θα ακούει στη θύρα 5634, θα δημιουργήσει 8 νήματα εργάτες, θα χρησιμοποιήσει ένα buffer που κρατά μέχρι 16 συνδέσεις που περιμένουν εξυπηρέτηση, και θα αποθηκεύει τις ψήψους των χρηστών στο pollLog.txt και τα αποτελέσματα της δημοσκόπησης στο αρχείο pollStats.txt.

Υλοποίηση
Ο εξυπηρετητής που θα υλοποιήσετε θα υποστηρίζει
- τη δημιουργία ενός νήματος-αρχηγού
- τη δημιουργία του buffer για την αποθήκευση των συνδέσεων
- την ενημέρωση του log αρχείου με τα στατιστικά της δημοσκόπησης

Master thread

Στην υλοποίηση σας, θα πρέπει να έχετε ένα νήμα-αρχηγό (master thread) που ξεκινά δημιουργώντας numWorkerThread νήματα-εργάτες. Το master thread θα δέχεται συνδέσεις από πελάτες με την **accept** κλήση συστήματος και θα τοποθετεί τους περιγραφείς αρχείων που αντιστοιχούν στις συνδέσεις σε έναν buffer συγκεκριμένου μεγέθους (που ορίζεται από το bufferSize). Το νήμα-αρχηγός ΔΕΝ θα διαβάζει από τις συνδέσεις που δέχεται. Απλώς, όποτε δέχεται κάποια σύνδεση θα τοποθετεί τον socket descriptor στον buffer και θα συνεχίζει να δέχεται επόμενες συνδέσεις.

Workers threads

Η δουλειά των νημάτων-εργατών είναι να διαβάζουν τις αιτήσεις από τα socket descriptors και να εξυπηρετούν τους πελάτες. Ένα νήμα-εργάτης ξυπνά όταν υπάρχει τουλάχιστον ένας descriptor στον buffer (δηλαδή τουλάχιστον ένας πελάτης έχει συνδεθεί στον εξυπηρετητή).
Όταν ξυπνά, το νήμα-εργάτης, στέλνει το μήνυμα “SEND NAME PLEASE” και περιμένει να διαβάσει από τον socket descriptor την απόκριση (το όνομα ενός ψηφοφόρου) από τον πελάτη.
Αν ο ψηφοφόρος έχει ήδη ψηφίσει, στέλνει απόκριση “ALREADY VOTED” και τερματίζει τη σύνδεση.
Αλλιώς στέλνει το μήνυμα “SEND VOTE PLEASE” και περιμένει να διαβάσει την απόκριση (μια συμβολοσειρά με το όνομα ενός κόμματος).
Θα τυπώνει σε μια γραμμή, στο τέλος του poll-log αρχείου, το όνομα του ψηφοφόρου και το κόμμα που ψήφισε. Τέλος στέλνει ένα μήνυμα “VOTE for Party XYZ RECORDED” (όπου το XYZ είναι το κόμμα που ψήφισε ο ψηφοφόρος) και τερματίζει τη σύνδεση.

Συγχρονισμός Master-workers μέσω του buffer.

Το νήμα-αρχηγός και τα νήματα-εργάτες έχουν σχέση παραγωγού-καταναλωτή και έτσι στην υλοποίηση σας θα πρέπει οι προσβάσεις τους στο κοινό buffer να συγχρονίζονται.
Συγκεκριμένα:

- το νήμα-αρχηγός πρέπει να μπλοκάρεται και να περιμένει όταν ο buffer είναι γεμάτος
- ένα νήμα-εργάτης πρέπει να περιμένει αν ο buffer είναι άδειος.

Με αυτή τη προσέγγιση, αν υπάρχουν περισσότερα νήματα-εργάτες από ενεργές συνδέσεις, τότε κάποια από τα νήματα-εργάτες θα μπλοκάρονται, περιμένοντας νέες συνδέσεις να φτάσουν στον εξυπηρετητή.

Σε αυτήν την εργασία, θα πρέπει να χρησιμοποιήσετε μεταβλητές συνθήκης στην υλοποίηση σας. **Αν η υλοποίηση σας κάνει οτιδήποτε busy-waiting, θα υπάρξει ποινή (-20 μονάδες).**

Εξυπηρετητής- Αποθήκευση Στατιστικών

Ο εξυπηρετητής θα πρέπει να κρατήσει κάποια στοιχεία για την δημοσκόπηση:

- τα ονόματα χρηστών που έχουν συμμετάσχει μέχρι στιγμής στη δημοσκόπηση μαζί με τις ψήφους τους, και
- τον αριθμό των χρηστών που έχουν ψηφίσει καθε κόμμα.

Στην υλοποίηση, θα πρέπει να αποθηκεύονται οι ψήφοι που λαμβάνει ο εξυπηρετητής στα αρχεία poll-log και poll-stats καθώς στέλνονται. Προσοχή εάν πάρει σήμα τερματισμού ο εξυπηρετητής θα πρέπει να γράψει τις ψήφους που έχει πάρει ως εκείνη τη στιγμή πριν τερματίσει ώστε το αποτέλεσμα να είναι σωστό. Μπορείτε να κρατήσετε αυτά τα στοιχεία σε οποιαδήποτε δομή δεδομένων σας εξυπηρετεί.

Καθώς τα νήματα που εξυπηρετούν τους χρήστες θα πρέπει να ενημερώνουν τα δεδομένα που κρατά ο server, **θα πρέπει να υλοποιήσετε και τον κατάλληλο συγχρονισμό ώστε να γίνονται τροποποιήσεις στα δεδομένα με σωστό τρόπο**.

Ασχέτως τον τρόπο που θα επιλεγεί για την ενημέρωση του αρχείου, η τελική τους μορφή θα έχει ώς εξής

αρχείο poll-log:

```
voter_name1 party_voted_for1
voter_name2 party_voted_for2
.
.
.
```

Ο εξυπηρετητής θα τρέχει μέχρι να λάβει ένα SIGINT (control-C) σήμα. Όταν λάβει SIGINT, θα γράφει τα αποτελέσματα της δημοσκόπησης στο αρχείο poll-stats με την ακόλουθη μορφή:

αρχείο poll-stats

```
party_name1   number_of_votes
party_name2   number_of_votes
party_name3   number_of_votes
.
.
.
TOTAL total_number_of_votes
```

Problem 2: Batch client for testing purposes (30 points) Γράψτε έναν πολυνηματικό πελάτη σε C/C++ που ονομάζεται pollSwayer που παίρνει τρία ορίσματα:

```
prompt> pollSwayer [serverName] [portNum] [inputFile.txt]
```

όπου:

- **serverName:** το όνομα του εξυπηρετητή στον οποίο θα συνδεθεί
- **portNum:** αριθμός θύρας που ακούει ο εξυπηρετητής
- **inputFile.txt:** ένα αρχείο που περιέχει ονόματα και ψήφους των ψηφοφόρων

Τα ορίσματα είναι όλα υποχρεωτικά και πρέπει να δίνονται με τη συγκεκριμένη σειρά και μόνο.

Για παράδειγμα, αν τρέξετε το πρόγραμμα σας με τα ακόλουθα ορίσματα:

```
prompt> pollSwayer linux01.di.uoa.gr 5634 inputFile.txt
```

Το `pollSwayer` θα διαβάζει το `inputFile.txt` και για κάθε γραμμή με όνομα, επώνυμο, και ψήφο, θα δημιουργεί ένα νήμα που θα συνδέεται στον εξυπηρετητή `linux01.di.uoa.gr` στο port 534 και θα του

στέλνει τη ψήφο ακολουθώντας το πρωτόκολλο που περιγράψαμε παραπάνω. Αν τρέξετε το πρόγραμμα αυτό από πολλαπλά διαφορετικά κελύφη συγχρόνως θα σας βοηθήσει να βρείτε και να διορθώσετε λάθη συγχρονισμού στον εξυπηρετητή σας.

Problem 3: Bash scripts (20 points). Θα γράψετε μια σειρά από bash scripts τα οποία θα χρησιμοποιήσετε για debugging του προγράμματός σας. Φυσικά, κατά τη διάρκεια της ανάπτυξης του προγράμματός σας μπορείτε να χρησιμοποιήσετε λίγα και μικρά αρχεία για να κάνετε debug. Το πρώτο script `create_input.sh` δουλεύει ως εξής (10 points):

```
./create_input.sh politicalParties.txt numLines
```

- `politicalParties.txt`: ένα αρχείο το οποίο περιέχει τα ονόματα όλων των κομμάτων στη δημοσκόπηση, ένα κόμμα ανά γραμμή.
- `numLines`: ο αριθμός γραμμών που θα περιέχει το `inputFile` που θα δημιουργεί το script.

Το script κάνει τα εξής

1. Κάνει ελέγχους για ορίσματα/νούμερα εισόδου.
2. Δημιουργεί ένα αρχείο `inputFile`.
3. Δημιουργεί `numLines` γραμμές στο `inputFile` όπου κάθε γραμμή περιέχει ένα τυχαίο όνομα και επώνυμο (τυχαίες συμβολοσειρές με τυχαίο μήκος από 3 έως 12 χαρακτήρες) και το όνομα ενός κόμματος (θα επιλέγει τυχαία το script κάποιο κόμμα από το `politicalParties.txt`).

Ίσως σας φανεί χρήσιμη η $RANDOM Bash function. (Δείτε, π.χ. http://tldp.org/LDP/abs/html/randomvar.html)

Το δεύτερο script `tallyVotes.sh` δουλεύει ως εξής (5 points):

```
./tallyVotes.sh tallyResultsFile
```

1. Κάνει έλεγχο αν υπάρχει ένα `inputFile` αρχείο στον τρέχοντα κατάλογο. Αν δεν υπάρχει ή δεν έχει τα κατάλληλα δικαιώματα χρήσης, τυπώνει ένα μήνυμα λάθους και τερματίζει.
2. Διαβάζει το αρχείο `inputFile` και μετράει τον αριθμο ψήφων που μαζεύει το κάθε κόμμα. Αν υπάρχει duplicate ψηφοφόρος, μετράει μόνο την πρώτη ψήφο του.
3. Σε ένα `tallyResultsFile` τοποθετεί τα αποτελέσματα όπου κάθε γραμμή περιέχει το όνομα ενός κόμματος και τον αριθμο ψήφων που συγκέντρωσε το κόμμα.

Το τρίτο script `processLogFile.sh` δουλεύει ως εξής (5 points):

```
./processLogFile.sh poll-log
```

1. Κάνει έλεγχο αν υπάρχει ένα `poll-log` αρχείο στον τρέχοντα κατάλογο. Αν δεν υπάρχει ή δεν έχει τα κατάλληλα δικαιώματα χρήσης, τυπώνει ένα μήνυμα λάθους και τερματίζει.
2. Διαβάζει το αρχείο `poll-log`. Για κάθε κόμμα, μετράει τον αριθμο ψήφων που εμφανίζονται στο `poll-log`. Αν υπάρχει duplicate ψηφοφόρος, μετράει μόνο την πρώτη ψήφο του.

3. Σε ένα `pollerResultsFile` τοποθετεί τα αποτελέσματα όπου κάθε γραμμή περιέχει το όνομα ενός κόμματος και τον αριθμο ψήφων που συγκέντρωσε το κόμμα.

Όταν θα τρέχετε τα scripts `tallyVotes.sh` και `processLogFile.sh`, θα πρέπει τα output files που παράγονται (`tallyResultsFile` και `pollerResultsFile`) να βγαίνουν ακριβώς τα ίδια και να ταιριάζουν πλήρως με τα αποτελέσματα που θα βγάζει ο `poller` στο αρχείο poll-stats όταν τερματίζει με SIGINT (control-C) σήμα.

Διαδικαστικά

● Το πρόγραμμά σας θα πρέπει να γραφεί σε C (ή C++) και θα πρέπει να τρέχει στα Linux workstations του Τμήματος. Κώδικας που δε μεταγλωττίζεται εκεί, θεωρείται ότι δεν μεταγλωττίζεται.

● Για επιπρόσθετες ανακοινώσεις, παρακολουθείτε το forum του μαθήματος στο piazza.com. Η πλήρης διεύθυνση είναι https://piazza.com/uoa.gr/spring2023/k24/home. Η παρακολούθηση του φόρουμ στο Piazza είναι υποχρεωτική.

● Ο κώδικάς σας θα πρέπει να αποτελείται από τουλάχιστον δύο (και κατά προτίμηση περισσότερα) διαφορετικά αρχεία. Η χρήση του separate compilation είναι επιτακτική και ο κώδικάς σας θα πρέπει να έχει ένα Makefile.

● Βεβαιωθείτε πως ακολουθείτε καλές πρακτικές software engineering κατά την υλοποίηση της άσκησης. Η οργάνωση, η αναγνωσιμότητα και η ύπαρξη σχολίων στον κώδικα αποτελούν κομμάτι της βαθμολογίας σας.

● Η υποβολή θα γίνει μέσω eclass.

● Ο κώδικάς σας θα πρέπει να κάνει compile στα εκτελέσιμα `poller` και `pollSwayer` όπως **ακριβώς** ορίζει η άσκηση και να έχει τις ίδιες παραμέτρους όπως ακριβώς ορίζει η άσκηση.

Τι πρέπει να παραδοθεί

● Όλη η δουλειά σας (πηγαίος κώδικας, Makefile και README) σε ένα tar.gz file με ονομασία OnomaEponymoProject2.tar.gz. Προσοχή να υποβάλετε μόνο κώδικα, Makefile, README και όχι τα binaries. Η άσκησή σας θα γίνει compile από την αρχή πριν βαθμολογηθεί.

● Όποιες σχεδιαστικές επιλογές κάνετε, θα πρέπει να τις περιγράψετε σε ένα README (απλό text αρχείο) που θα υποβάλλετε μαζί με τον κώδικά σας. Το README χρειάζεται να περιέχει μια σύντομη και περιεκτική εξήγηση για τις επιλογές που έχετε κάνει στον σχεδιασμό του προγράμματός σας. 1-2 σελίδες ASCII κειμένου είναι αρκετές. Συμπεριλάβετε την εξήγηση και τις οδηγίες για το compilation και την εκτέλεση του προγράμματός σας.

● Ο κώδικας που θα υποβάλετε θα πρέπει να είναι δικός σας. Απαγορεύεται η χρήση κώδικα που δεν έχει γραφεί από εσάς ή κώδικας που έχει γραφτεί με τη βοήθεια μηχανών τύπου chatGPT.

● Καλό θα είναι να έχετε ένα backup .tar της άσκησής σας όπως ακριβώς αυτή υποβλήθηκε σε κάποιο εύκολα προσπελάσιμο μηχάνημα (server του τμήματος, github, cloud).

● Η σωστή υποβολή ενός σωστού tar.gz που περιέχει τον κώδικα της άσκησής σας και ό,τι αρχεία χρειάζονται είναι αποκλειστικά ευθύνη σας. **Άδεια tar/tar.gz ή tar/tar.gz που έχουν λάθος και δε γίνονται extract δε βαθμολογούνται**.

Τι θα βαθμολογηθεί

- Η συμμόρφωση του κώδικά σας με τις προδιαγραφές της άσκησης.
- Η οργάνωση και η αναγνωσιμότητα (μαζί με την ύπαρξη σχολίων) του κώδικα.
- Η χρήση Makefile και το separate compilation.

Άλλες σημαντικές παρατηρήσεις

- Οι εργασίες είναι ατομικές.
- Όποιος υποβάλλει / δείχνει κώδικα που δεν έχει γραφτεί από την ίδια/τον ίδιο **μηδενίζεται** στο μάθημα.
- Αν και αναμένεται να συζητήσετε με φίλους και συνεργάτες το πώς θα επιχειρήσετε να δώσετε λύση στο πρόβλημα, αντιγραφή κώδικα (οποιασδήποτε μορφής) είναι κάτι που δεν επιτρέπεται. Οποιοσδήποτε βρεθεί αναμεμειγμένος σε αντιγραφή κώδικα απλά παίρνει μηδέν στο μάθημα. Αυτό ισχύει για όσους εμπλέκονται ανεξάρτητα από το ποιος έδωσε/πήρε κλπ. Τονίζουμε πως θα πρέπει να λάβετε τα κατάλληλα μέτρα ώστε να είναι προστατευμένος ο κώδικάς σας και να μην αποθηκεύεται κάπου που να έχει πρόσβαση άλλος χρήστης (π.χ., η δικαιολογία «Το είχα βάλει σε ένα github repo και μάλλον μου το πήρε από εκεί», δεν είναι δεκτή.)
- Οι ασκήσεις προγραμματισμού μπορούν να δοθούν με καθυστέρηση το πολύ 3 ημερών και με ποινή 5% για κάθε μέρα αργοπορίας. Πέραν των 3 αυτών ημερών, δεν μπορούν να κατατεθούν ασκήσεις.
- Το πρόγραμμά σας θα πρέπει να γραφτεί σε C ή C++. Μπορείτε να χρησιμοποιήσετε μόνο εντολές οι οποίες είναι διαθέσιμες στα μηχανήματα Linux του τμήματος. Πρόγραμμα που πιθανόν μεταγλωτίζεται ή εκτελείται στο προσωπικό σας υπολογιστή αλλά όχι στα μηχανήματα Linux του τμήματος, θεωρείται ότι δε μεταγλωτίζεται ή εκτελείται αντίστοιχα.