

Σκοπός

Στην άσκηση αυτή θα εξοικειωθείτε με τη δημιουργία διεργασιών με χρήση των κλήσεων συστήματος `fork/exec`, την επικοινωνία διεργασιών μέσω `pipes` και `named pipes`, τη χρήση `low-level I/O`, τον χειρισμό `signals` και τη δημιουργία `shell scripts`.

Ζητούμενο 1 (βάρος: 85%)

Οι βασικές οντότητες της εργασίας είναι ο `listener`, ο `manager`, και οι `workers`.

Listener: Η εντολή `inotifywait` στέλνει ειδοποιήσεις για τις αλλαγές στα περιεχόμενα ενός καταλόγου του `file system`. Με χρήση της `inotifywait` θα κάνετε `monitoring` των αλλαγών στα αρχεία ενός `directory`. Η `inotifywait` θα εκτελεστεί (με κλήση της οικογένειας `exec`) μέσα σε μια δική σας διεργασία `listener`.

Manager: Αποτελεί την κεντρική οντότητα του συστήματος η οποία επικοινωνεί με τη διεργασία `listener` μέσω **pipe**. Ο `listener` θα ενημερώνει τον `manager` για κάθε νέο αρχείο που ανιχνεύει στο `directory` που παρακολουθεί. Δηλαδή, πριν εκτελέσετε την `inotifywait`, θα πρέπει να έχετε συνδέσει την έξοδο της διεργασίας με το `pipe`.

Workers: Ο `manager` επικοινωνεί με τους `workers` μέσω **named pipes**. Για κάθε όνομα αρχείου που λαμβάνει ο `manager` από τον `listener`, θα ειδοποιεί ή θα δημιουργεί (αν δεν υπάρχει) μια διεργασία `worker`, η οποία ενεργεί πάνω στο συγκεκριμένο αρχείο. Κατά την έναρξη, ο `manager` δημιουργεί τόσους `workers` όσα και τα αρχεία που ενημερώνεται ότι υπάρχουν. Ο `manager` θα πρέπει να κρατάει πληροφορία για τους διαθέσιμους `workers` σε μία ουρά και να δημιουργεί νέους `workers` μόνο αν δεν υπάρχει κανείς διαθέσιμος. Ο κάθε `worker` αναλαμβάνει να επεξεργαστεί ένα αρχείο τη φορά. Τα ονόματα των `named pipes` είναι στην ευχέρειά σας.

Λειτουργία

Ο `manager`, αν ο `worker` είναι σταματημένος, πρέπει να τον ξεκινήσει (με `signal SIGCONT`, επειδή οι διαθέσιμοι `workers` θα είναι σε κατάσταση “`stopped`”) και θα στείλει στον `worker` πληροφορία για το ποιο αρχείο να επεξεργαστεί.

Σκοπός του `worker` είναι να ανοίξει το αρχείο και να αναζητήσει `urls` μέσω `low-level I/O`. Τα αρχεία είναι αρχεία κειμένου που μπορεί να περιέχουν απλό κείμενο και `URLs`. Η αναζήτηση περιορίζεται στα `URLs` που χρησιμοποιούν το πρωτόκολλο **http**, δηλαδή της μορφής `http://...`. Πιο συγκεκριμένα, το κάθε `URL` ξεκινάει με `http://` και τελειώνει με έναν κενό χαρακτήρα.

Για κάθε `URL` που εντοπίζεται, απαιτείται να εξαχθεί η πληροφορία για το `location` του, χωρίς το `www`. Δείτε το ακόλουθο `link`:

<https://www.techopedia.com/definition/1352/uniform-resource-locator-url>

για τα διαφορετικά μέρη ενός `URL`. Εμείς, όπως είπαμε, ενδιαφερόμαστε να βρούμε το

location, χωρίς το www, αν υπάρχει. Για παράδειγμα, για το URL της ιστοσελίδας του τμήματος <http://www.di.uoa.gr/> ως location έχουμε το “di.uoa.gr”.

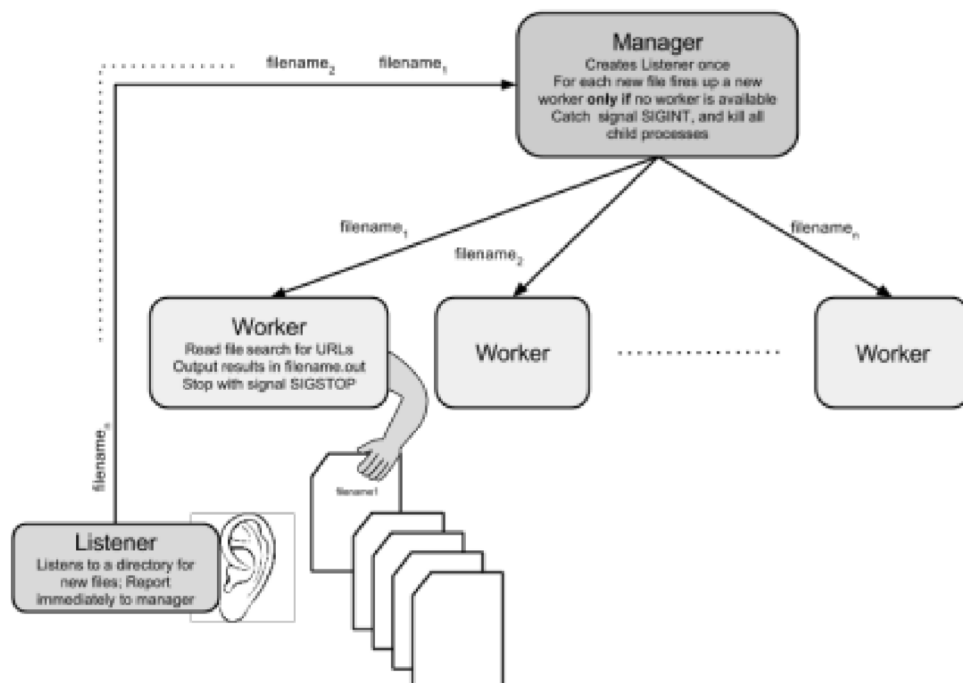
Κατά τη διάρκεια της ανάγνωσης του αρχείου, ο `worker` δημιουργεί ένα νέο αρχείο στο οποίο καταγράφει όλα τα `locations` που ανίχνευσε μαζί με τον αριθμό εμφάνισής τους. Π.χ. αν στο αρχείο που προστέθηκε εμφανίζονται 3 URLs με `location` “`di.uoa.gr`”, το αρχείο εξόδου του `worker` θα περιέχει τη γραμμή “`di.uoa.gr 3`” και αντίστοιχα μια γραμμή για κάθε άλλο `location`.

Αν το αρχείο που διάβασε ο worker έχει όνομα `<filename>`, το αρχείο που δημιουργεί έχει όνομα `<filename>.out`.

Στη συνέχεια, ο `worker` ειδοποιεί τον `manager` ότι τελείωσε την εργασία του και είναι διαθέσιμος για επόμενη εντολή. Ο `worker` στέλνει στον εαυτό του `signal SIGSTOP` έτσι ώστε να μπει σε κατάσταση “`stopped`” και ο `manager`, που είναι γονιός της διεργασίας `worker`, να λάβει σήμα `SIGCHLD` και να δει ποιο παιδί άλλαξε κατάσταση με `waitpid()`.

Οι διεργασίες δεν τερματίζουν από μόνες τους πρέπει να τις σταματήσει ο χρήστης. Ο τερματισμός του manager γίνεται με Control+C (σήμα SIGINT), και πριν τερματίσει πρέπει να σκοτώνει όλες τις υπόλοιπες διεργασίες (listener και workers).

Η δομή της ιεραρχίας των διεργασιών συνοψίζεται στο Σχήμα 1.



Σχήμα 1

Ζητούμενο 2 (βάρος: 15%)

Έχοντας υλοποιήσει το ζητούμενο 1, έχετε δημιουργήσει έναν αριθμό αρχείων της μορφής `<filename>.out`, τα οποία περιέχουν τα domains που βρέθηκαν μαζί με τον αριθμό εμφάνισής τους.

Σκοπός σας σε αυτό το σημείο είναι να φτιάξετε ένα shell script `finder.sh`, το οποίο θα δέχεται ως όρισμα ένα ή περισσότερα Top Level Domain (TLD) που θέλετε να αναζητήσετε στο σύνολο των `.out` αρχείων. Συγκεκριμένα, θα πρέπει να βρείτε το συνολικό πλήθος των εμφανίσεων του TLD στο σύνολο των αρχείων που δημιουργήσατε.

Τα αρχεία σας από το προηγούμενο ερώτημα είναι της μορφής:

```
location num_of_appearances
```

Για παράδειγμα, δίνοντας ως όρισμα το TLD “com” το αποτέλεσμα θα είναι το άθροισμα των “num_of_appearances” όπου τα locations τελειώνουν σε com.

Εκτέλεση Προγράμματος

Το πρόγραμμα σας θα ονομάζεται `sniffer`.

Η εκτέλεση του θα γίνεται **αυστηρά** ως εξής:

```
./sniffer [-p path]
```

Η προαιρετική παράμετρος “-p path” χρησιμοποιείται για να υποδείξει το path του directory που θέλουμε να γίνει το monitoring.

Στην περίπτωση που δε δίνεται αυτή η επιλογή λαμβάνεται ως default ο τρέχων κατάλογος. Το πρόγραμμα θα πρέπει να εκτελείται απρόσκοπτα χωρίς περαιτέρω αλληλεπίδραση από τον χρήστη (όχι χρήση διαφόρων μενού κτλ),

Λεπτομέρειες Υλοποίησης

- Η διαχείριση των αρχείων πρέπει να γίνει αποκλειστικά με low-level IO.
- Η επικοινωνία μεταξύ των διεργασιών θα γίνεται αποκλειστικά με pipes, named pipes και signals όπως ακριβώς προσδιορίζεται παραπάνω.
- Η εντολή `inotifywait` είναι εγκατεστημένη στα μηχανήματα του τμήματος και μπορείτε να την εγκαταστήσετε και σε δικό σας μηχανήμα. Λεπτομέρειες για την εντολή συστήματος `inotifywait` μπορείτε να βρείτε στο manual της εντολής.
- Μπορείτε να θεωρήσετε ότι δε θα υπάρξει καμία αλλαγή στα αρχεία τα οποία εισάγονται στο directory.
- Υπόδειξη: Ο `manager` όταν πάρει σήμα από `worker` (για να ειδοποιηθεί ότι ο `worker` είναι ξανά διαθέσιμος) ξεμπλοκάρει από την κλήση `read` που κάνει στο κανάλι επικοινωνίας με τον `listener`. Το `read` επιστρέφει τιμή που δείχνει ότι διακόπηκε. Μπορεί τότε να ανανεώσει την πληροφορία του για διαθέσιμους `worker` και να επαναλάβει το `read`.

- Προσοχή: Το output των workers θα πρέπει να είναι σε κάποιο directory το οποίο δεν επηρεάζει τον `listener`, ώστε να μην “ακούει” για τα αρχεία εξόδου. Θα πρέπει κατά την εκτέλεση του προγράμματος να αδειάζετε πιθανά αρχείο εξόδου από προηγούμενη εκτέλεση.

Διαδικαστικά

- Το πρόγραμμά σας θα πρέπει να τρέχει στα μηχανήματα Linux/Unix της σχολής. 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/spring2022/k24/home>. Η παρακολούθηση του φόρουμ στο Piazza είναι υποχρεωτική.
- Το πρόγραμμά σας θα πρέπει να γραφεί σε C (ή C++). Σε κάθε περίπτωση το πρόγραμμά σας θα πρέπει να τρέχει στα Linux workstations του Τμήματος.
- 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 στο εκτελέσιμο `sniffer` όπως ακριβώς ορίζει η άσκηση και να έχει την ίδια παράμετρο `-p` όπως ακριβώς ορίζει η άσκηση. Το ίδιο ισχύει και για το shell script το οποίο θα πρέπει να ονομάζεται `finder.sh`.

Τι πρέπει να παραδοθεί

- Όλη η δουλειά σας (πηγαίος κώδικας, Makefile και README) σε ένα tar.gz file με ονομασία `OnomaEponymoProject1.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 υποβάλετε 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.
- Καλό θα είναι να έχετε ένα backup .tar της άσκησής σας όπως ακριβώς αυτή υποβλήθηκε σε κάποιο εύκολα προσπελάσιμο μηχανήμα (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 αναμειγμένος 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++ για το 1ο ζητούμενο και bash για τον 2ο ζητούμενο. Μπορείτε να χρησιμοποιήσετε μόνο εντολές οι οποίες είναι διαθέσιμες στα μηχανήματα Linux του τμήματος.