Άσκηση 2

Προγραμματισμός Συστήματος

Προθεσμία Παράδοσης: 25 Απριλίου 2015

Σκοπός

Στην άσκηση αυτή θα εξοικειωθείτε με την δημιουργία διεργασιών με χρήση των κλήσεων συστήματος fork/exec, την επικοινωνία διεργασιών μέσω pipes/named pipes, τη χρήση low-level I/0, το χειρισμό signals και τη δημιουργία shell scripts.

Ζητούμενο 1 (βάρος: 85%)

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. Μια διεργασία-manager θα επικοινωνεί (πρόταση: μέσω pipe) με την διεργασία-listener, η οποία θα την ενημερώνει για κάθε νέο αρχείο που ανιχνεύει στο directory που παρακολουθεί. (Δηλαδή πριν εκτελέσετε inotifywait, θα πρέπει να έχετε συνδέσει την εξοδό της διεργασίας με το pipe.)

Για κάθε όνομα αρχείου που λαμβάνει ο manager, θα ειδοποιεί ή δημιουργεί μια διεργασία-worker, η οποία ενεργεί πάνω στο συγκεκριμένο αρχείο. Ο manager θα πρέπει να κρατάει πληροφορία για τους διαθέσιμους workers (αρχικά δεν υπάρχουν) και να δημιουργεί νέο worker μόνο αν δεν υπάρχει κανείς διαθέσιμος. Ο manager αν ο worker είναι σταματημένος πρέπει να τον ξεκινήσει (πρόταση: με signal SIGCONT, επειδή οι διαθέσιμοι workers θα είναι σε κατάσταση “stopped”) και θα στείλει στον worker πληροφορία για το ποιο αρχείο να επεξεργαστεί (πρόταση: μέσω named pipe).

Σκοπός του worker είναι να ανοίξει το αρχείο και να αναζητήσει urls μέσω low-level I/O. Τα αρχεία δεν είναι αναγκαστικά αρχεία κειμένου - μπορεί να βρεθούν URLs και μέσα σε binary αρχεία. Η αναζήτηση περιορίζεται στα urls που χρησιμοποιούν το πρωτόκολλο http, δηλαδή της μορφής http://… . Για κάθε url που εντοπίζεται, απαιτείται να εξαχθεί η πληροφορία για το domain του. Ως domain ορίζουμε τα δύο τελευταία μέρη του url, δηλαδή τις δύο λέξεις που χωρίζονται με “.” ακριβώς πριν το πρώτο “/” μετά το “http://”. Για παράδειγμα, για το url της ιστοσελίδας του τμήματος http://www.di.uoa.gr/ ως domain έχουμε το “uoa.gr”.

Κατά την διάρκεια της ανάγνωσης του αρχείου, ο worker δημιουργεί ένα νέο αρχείο στο οποίο καταγράφει όλα τα domains που ανίχνευσε μαζί με τον αριθμό εμφάνισής τους. Π.χ. αν στο αρχείο που προστέθηκε εμφανίζονται 3 URLs με domain “uoa.gr”, το αρχείο εξόδου του worker θα περιέχει τη γραμμή “uoa.gr 3” και αντίστοιχα για κάθε άλλο domain. Αν το αρχείο που διάβασε ο worker έχει όνομα <*filename*>, το αρχείο που δημιουργεί έχει όνομα <*filename*>.out.

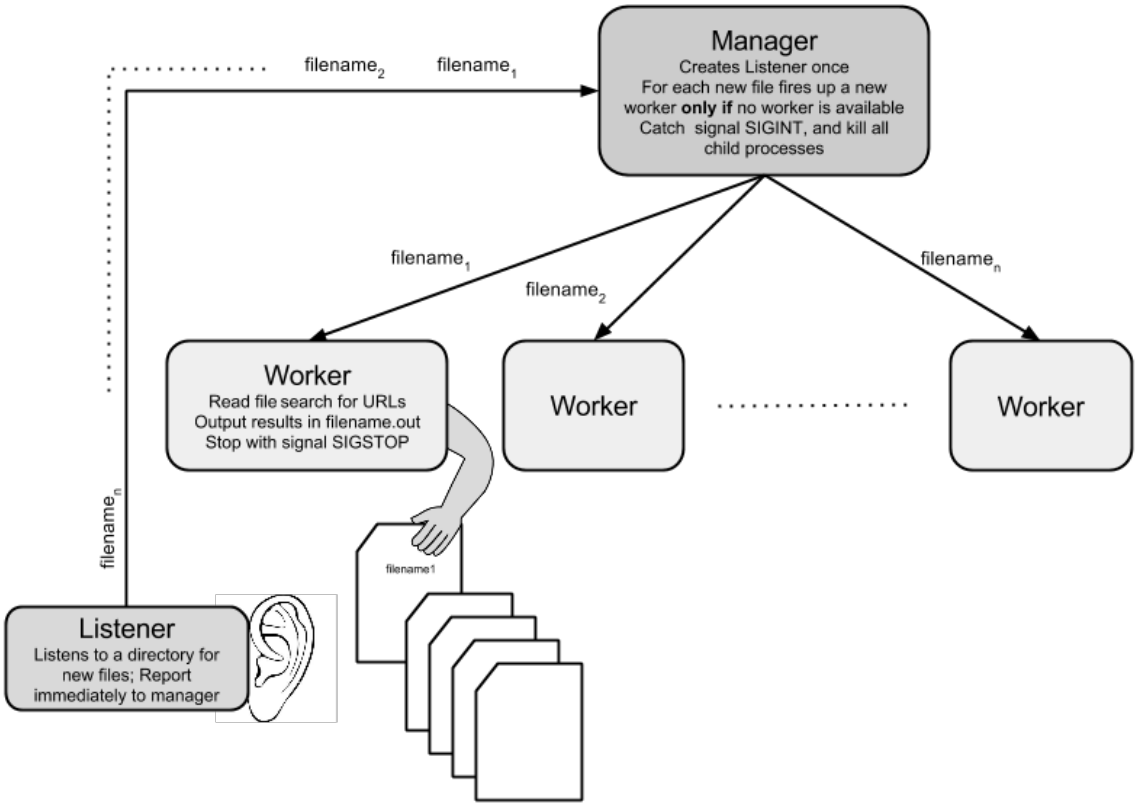
Στη συνέχεια, ο worker ειδοποιεί τον manager ότι τελείωσε την εργασία του και είναι διαθέσιμος για επόμενη εντολή. (Πρόταση: ο worker στέλνει στον εαυτό του signal SIGSTOP έτσι ώστε να μπει σε κατάσταση “stopped” και ο manager, που είναι γονιός της διεργασίας worker, να λάβει σήμα SIGCHLD και να δει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

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, το οποίο θα δέχεται ως όρισμα ένα ή περισσότερα Top Level Domain (TLD) που θέλετε να αναζητήσετε στο σύνολο των .out αρχείων. Συγκεκριμένα, έχοντας αρχεία με εγγραφές της μορφής
“xx.yy num_of_appearances”, και δίνοντας ως όρισμα το tld “com” πρέπει να βρείτε πόσες φορές στο σύνολο των αρχείων που δημιουργήσατε εμφανίζεται το συγκεκριμένο tld, δηλαδή πόσες φορές έχετε “yy”=”com”. (Πρόταση: μπορεί να σας φανούν χρήσιμες οι εντολές του shell στην πρώτη ενότητα διαφανειών.)



Σχήμα 1

Εκτέλεση Προγράμματος

Έστω οτι το προγραμμα σας ονομάζεται sniffer. Η εκτέλεση του θα γίνεται 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 δεν δίνεται αυτή η επιλογή λαμβάνεται ως default ο τρέχων κατάλογος.

Λεπτομέρειες Υλοποίησης

Λεπτομέρειες για την εντολή συστήματος inotifywait μπορείτε να βρείτε στο manual της εντολής.

- Η διαχείριση των αρχείων πρέπει να γίνει αποκλειστικά με low-level IO.
- Η επικοινωνία μεταξύ των διεργασιών θα γίνεται αποκλειστικά με pipes, named pipes (FIFO queues) και signals.
- Η εντολή inotifywait είναι εγκατεστημένη στα μηχανήματα του τμήματος και μπορείτε να την εγκαταστήσετε και σε δικό σας μηχάνημα. Αν θέλετε να δουλέψετε σε MacOS, υπάρχουν αντίστοιχα εργαλεία, που είναι δική σας ευθύνη να τα βρειτε. Μπορείτε επίσης να φτιάξετε ένα αντίστοιχο πρόγραμμα του inotifywait, αλλά με δικιά σας ευθύνη.
- Μπορείτε να θεωρήσετε ότι δεν θα υπάρξει καμία αλλαγή στα αρχεία τα οποια εισάγονται στο directory.
- Υπόδειξη: Ο manager όταν πάρει σήμα από worker (για να ειδοποιηθεί ότι ο worker είναι ξανά διαθέσιμος) ξε-μπλοκάρει 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.

Διαδικαστικά

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

Τι πρέπει να παραδοθεί

Όλη η δουλειά σας σε ένα tar-file που να περιέχει όλα τα source files, header files, Makefile.

- 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).

- 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

- 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 η κομματιαστή σύμβολο-μετάφραση (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 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.
- 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.