Τμήμα Μηχ. Η/Υ & Πληροφορικής **Διδάσκων: Στέργιος Αναστασιάδης**
Πανεπιστήμιο Ιωαννίνων **Εαρινό Εξάμηνο 2021**

ΜΥΥ601 Λειτουργικά Συστήματα

Εργαστηριακή Άσκηση 2

Υλοποίηση αρχείου καταγραφής στο σύστημα αρχείων FAT του Linux

Ανακοίνωση: Παρασκευή, 9 Απριλίου, 2021, Προθεσμία: Τρίτη, 11 Μαΐου 2021

Βαρύτητα 2ης Άσκησης: 20/30

1. Εισαγωγή

Σας δίνεται ο πηγαίος κώδικα του πυρήνα του Linux με την μορφή βιβλιοθήκης. Διασυνδέοντας την βιβλιοθήκη με μια εφαρμογή μπορείτε να τρέξετε τον πυρήνα του Linux σε επίπεδο χρήστη. Στην παρούσα άσκηση θα επεκτείνετε το σύστημα αρχείων FAT στην βιβλιοθήκη του πυρήνα Linux προκειμένου να παρέχετε αρχείο καταγραφής (journaling) που αποθηκεύει όλες τις τροποποιήσεις που συμβαίνουν στο σύστημα αρχείων κατά την εκτέλεση εφαρμογών.

2. Η βιβλιοθήκη Linux Kernel Library

Η βιβλιοθήκη Linux Kernel Library (LKL) είναι μια υλοποίηση του πυρήνα του Linux που εκτελείται σε επίπεδο χρήστη. Αναπτύχθηκε προκειμένου να διευκολύνει την επαναχρησιμοποίηση με ελάχιστη προσπάθεια του κώδικα του πυρήνα του Linux σε εφαρμογές. Η LKL μπορεί να χρησιμοποιηθεί σε διάφορα λειτουργικά συστήματα σε επίπεδο χρήστη ή πυρήνα. Είναι υλοποιημένη ως μια τροποποίηση του πυρήνα του Linux για μια εικονική αρχιτεκτονική υπολογιστή που ονομάζεται lkl. Οι εφαρμογές αλληλεπιδρούν με την LKL χρησιμοποιώντας την διεπαφή του πυρήνα του Linux.

Δεδομένου ότι η LKL τρέχει μέσα σε διεργασία, επιτρέπει αρκετές απλοποιήσεις του πυρήνα. Ειδικότερα, η LKL δεν χρειάζεται να υποστηρίζει πολλαπλές διεργασίες ή χώρους διευθύνσεων διεργασιών. Επιπλέον, δεν παρέχει προστασία μεταξύ επιπέδου χρήστη και πυρήνα. Η LKL δεσμεύει μια περιοχή μνήμης από την διεργασία για να χειριστεί την εκχώρηση μνήμης που σχετίζεται με ενδιάμεση αποθήκευση (buffers) και δυναμικές δομές δεδομένων.

Η LKL απαιτεί νήματα πυρήνα για να επεξεργαστεί τις αιτήσεις εισόδου/εξόδου και να χειριστεί τις διακοπές. Επομένως, χρειάζεται από την εφαρμογή (π.χ., Pthreads) τις βασικές λειτουργίες για να δημιουργήσει και να τερματίσει νήματα. Με παρόμοιο τρόπο, η εφαρμογή παρέχει σημαφόρους (με αρχική τιμή 0) που επιτρέπουν κάθε νήμα να μπαίνει σε αποκλεισμό όταν δημιουργείται και να ξεκινά την εκτέλεση όταν επιλέγεται από τον χρονοδρομολογητή της LKL.

Μια εφαρμογή μπορεί να αλληλεπιδράσει με τις τοπικές συσκευές μέσω της LKL, π.χ., να διαβάσει δεδομένα από ένα δίσκο. Σε αυτή την περίπτωση, η LKL παρέχει έναν οδηγό συσκευής μπλοκ που επικοινωνεί με τον οδηγό συσκευών του λειτουργικού συστήματος πάνω στο οποίο τρέχει. Όταν μια αίτηση εισόδου/εξόδου ολοκληρωθεί από μια τοπική συσκευή, το λειτουργικό σύστημα ενημερώνει την LKL χρησιμοποιώντας την υποστήριξη αιτήσεων διακοπών (IRQ). Η LKL διατηρεί μια ουρά με αιτήσεις διακοπών που το νήμα της εξυπηρετεί σειριακά.

Όταν κανένα νήμα δεν είναι διαθέσιμο να τρέξει, συνήθως ο πυρήνας του Linux εκτελεί ένα αδρανές νήμα που μειώνει την ταχύτητα εκτέλεσης του επεξεργαστή με ειδικές εντολές υλικού. Αντίθετα, η LKL βασίζεται σε σημαφόρους της εφαρμογής για να εισέλθει σε αποκλεισμό όταν δεν υπάρχει τίποτε να εκτελέσει μέχρι να ξυπνήσει όταν για τον χειρισμό κάποιας διακοπής.

Το API που παρέχει η LKL στην εφαρμογή είναι ένα υποσύνολο των κλήσεων συστήματος του Linux. Παρόλο που μια εφαρμογή μπορεί να αλληλεπιδράσει απευθείας με όλα τα σύμβολα της διασυνδεδεμένης LKL, είναι προτιμότερο να χρησιμοποιεί τις κλήσεις συστήματος με την παρεχόμενη προστασία τους. Εφόσον η LKL δεν υποστηρίζει παράλληλη επεξεργασία (SMP), οι κλήσεις συστήματος σειριοποιούνται στους χειριστές των διακοπών. Μια κλήση συστήματος μεταφέρεται στην LKL μέσω ενός wrapper που τοποθετεί την αίτηση σε ουρά (work queue). Όταν ολοκληρώνεται ο χειρισμός της κλήσης συστήματος, το νήμα της εφαρμογής βγαίνει από τον αποκλεισμό μέσω μιας σημαφόρου.

Ο πηγαίος κώδικας της LKL είναι οργανωμένος σε πολλαπλούς καταλόγους όπως και ο πυρήνας του Linux. Για την παρούσα άσκηση, μας ενδιαφέρει κυρίως ο κατάλογος **arch/lkl** που περιέχει τις εξαρτήσεις από την αρχιτεκτονική, ο κατάλογος **tools/lkl** που παρέχει τις τροποποιήσεις για να τρέξει ο πυρήνας του Linux ως βιβλιοθήκη, ο κατάλογος **fs/fat** που περιέχει την υλοποίηση του συστήματος αρχείων FAT και ο κατάλογος **mm** που υλοποιεί τις μεταφορές μεταξύ μνήμης και δίσκου.

Ο κατάλογος **fs** υλοποιεί διάφορα συστήματα αρχείων συμπεριλαμβανομένου του ext4 που είναι το default του Linux. Το σύστημα αρχείων ext4 είναι ένα παράδειγμα συστήματος αρχείων με αρχείο καταγραφής (journaling) που υποστηρίζει την καταγραφή των τροποποιήσεων για γρήγορη ανάκτηση από σφάλματα. Η καταγραφή τροποποιήσεων είναι μια εσωτερική λειτουργία που καταγράφει όλες τις αλλαγές στα δεδομένα και μεταδεδομένα του συστήματος αρχείων. Το αρχείο καταγραφής αποθηκεύεται είτε ως τοπικό αρχείο στο ext4 ή σε ξεχωριστή διαμέριση του δίσκου. Η καταγραφή τροποποιήσεων υλοποιείται με το ειδικό σύστημα αρχείων jbd2 που διαχειρίζεται την ακολουθία των τροποποιήσεων σε ομάδες που ονομάζονται συναλλαγές. Η καταγραφή τροποποιήσεων μπορεί να ρυθμιστεί να περιλαμβάνει μόνο μεταδεδομένα (π.χ., inodes) ή δεδομένα και μεταδεδομένα (π.χ., μπλοκ δεδομένων αρχείων). Ο κύριος ρόλος της καταγραφής τροποποιήσεων είναι η δυνατότητα να επαναλάβει τις καταγεγραμμένες τροποποιήσεις μετά από κάποια πτώση (crash) του συστήματος που δεν το επέτρεψε να μεταφέρει τις πρόσφατες τροποποιήσεις εγκαίρως από την μνήμη στον δίσκο.

3. Το σύστημα αρχείων FAT

Ένα σύστημα αρχείο στο Unix παρέχει ιεραρχική αποθήκευση δεδομένων μέσω τεσσάρων αφαιρέσεων: mount points, inodes, directory entries και files. Ένα σύστημα αρχείων είναι προσβάσιμο από συγκεκριμένο μονοπάτι (mount point) του ενοποιημένου δέντρου αρχείων της διεργασίας. Ένα αρχείο είναι μια αριθμημένη ακολουθία bytes. Τα αρχεία είναι οργανωμένα σε καταλόγους που περιέχουν σχετικά αρχεία ή άλλους καταλόγους. Κάθε συστατικό ενός μονοπατιού καλείται directory entry, ή dentry. Οι πληροφορίες που σχετίζονται με ένα αρχείο (π.χ., άδειες πρόσβασης, μέγεθος, αναγνωριστικό ιδιοκτήτη, χρόνος δημιουργίας, κλπ) ονομάζονται μεταδεδομένα αρχείων και συνήθως είναι αποθηκευμένα σε μια ξεχωριστή δομή δεδομένων που ονομάζεται inode. Τα μεταδεδομένα του συστήματος αρχείων αποτελούνται από πληροφορίες ελέγχου για το σύστημα αρχείων και είναι αποθηκευμένα σε μια ειδική δομή δεδομένων που ονομάζεται superblock.

Όπως στους περισσότερους πυρήνες του Unix, ο πυρήνας του Linux χρησιμοποιεί το Virtual File System (VFS) προκειμένου να εξάγει ένα ενοποιημένο, λογικό API στις εφαρμογές επιπέδου χρήστη για τα διάφορα συστήματα αρχείων και συσκευές που υποστηρίζει. Το API παρέχει συγκεκριμένες συναρτήσεις που είναι ανεξάρτητες από το σύστημα αρχείων ή τον δίσκο.

Το Linux υλοποιεί το σύστημα αρχείων FAT από τις παλιότερες εκδόσεις του. Σήμερα, τα συστήματα αρχείων FAT χρησιμοποιούνται συχνά για εξαγόμενα μέσα αποθήκευσης, όπως κάρτες μνήμης (π.χ,, έξυπνα κινητά, ψηφιακές φωτογραφικές μηχανές) και οδηγούς μνήμης USB (π.χ., PCs). Το σύστημα αρχείων FAT οργανώνει τον αποθηκευτικό χώρο στην reserved area, τα allocation tables, το root directory και την data area. Το σύστημα VFAT είναι ένα σύστημα FAT που υποστηρίζει ονόματα αρχείων μέχρι 255 χαρακτήρες αντί για σύντομα ονόματα αρχείων των προηγούμενων συστημάτων FAT (μορφή 8.3).

Η reserved area συνήθως περιέχει τον boot sector με πληροφορίες που σχετίζονται με το μέσο αποθήκευσης και την οργάνωσή σε σύστημα αρχείων. Ο δίσκος είναι οργανωμένος σε clusters που περιέχουν συγκεκριμένο πλήθος από συνεχόμενους τομείς (sectors). Ό αριθμός ενός cluster μπορεί να μετατραπεί στον αντίστοιχο αριθμό μπλοκ όπου ξεκινά στον δίσκο πολλαπλασιάζοντας το cluster με το πλήθος των τομέων ανά cluster και την απαραίτητη διόρθωση εφόσον οι δύο πρώτες εγγραφές του FAT δεν χρησιμοποιούνται.

Το File Allocation Table (FAT) περιέχει μία εγγραφή ανά cluster με μέγεθος εγγραφής ίσο με 12, 16 ή 32 bits για το FAT12, FAT16 και FAT32, αντίστοιχα. Μια εγγραφή FAT προσδιορίζει έναν δείκτη στο επόμενο cluster του αρχείου ή σηματοδοτεί το τρέχον cluster ως τελευταίο. Το σύστημα

FAT συνήθως περιέχει δύο αντίγραφα του FAT προκειμένου να παραμείνει χρησιμοποιήσιμο ακόμη και αν ένα από τα δύο καταστραφεί. Το root directory καταλαμβάνει μια συνεχόμενη ομάδα από clusters όπως προσδιορίζεται από την μορφοποίηση του συστήματος. Οι υπόλοιποι κατάλογοι είναι οργανωμένοι όπως και τα κανονικά αρχεία ως μια διασυνδεδεμένη λίστα από clusters που συνήθως δεν είναι συνεχόμενα στον δίσκο. Μια εγγραφή καταλόγου (dentry) περιέχει το όνομα αρχείου, την θέση των δεδομένων του αρχείου, και το μέγεθος του αρχείου σε bytes. Η μορφή είναι ίδια για το root directory ή έναν υποκατάλογο. Μια εγγραφή καταλόγου μπορεί να βρεθεί με απλή γραμμική αναζήτηση στον κατάλογο για συγκεκριμένο όνομα αρχείου.

Μια προσπέλαση αρχείου απαιτεί τον προσδιορισμό της λίστας των clusters που καταλαμβάνει στο δίσκο. Δεδομένου ότι μια εγγραφή καταλόγου έχει περιορισμένο μέγεθος που δεν χωράει όλη την λίστα των clusters, προσδιορίζει μόνο το πρώτο cluster του αρχείου. Τα υπόλοιπα clusters του αρχείου μπορούν να βρεθούν από την αντίστοιχη εγγραφή του FAT και ακολουθώντας τον αντίστοιχο δείκτη στο επόμενο cluster επαναληπτικά μέχρι το τέλος του αρχείου.

Η υλοποίηση του FAT στο Linux προσδιορίζει διάφορες λειτουργίες για την διαχείριση του superblock, των inodes, των αρχείων και των καταλόγων. Οι λειτουργίες του superblock ορίζονται στην δομή **struct super_operations fat_sops** του αρχείου **fs/fat/inode.c**. Οι λειτουργίες της μνήμης (χώρου διευθύνσεων) ορίζονται στην δομή **struct address_space_operations fat_aops** του αρχείου **fs/fat/inode.c**. Οι λειτουργίες των εγγραφών FAT ορίζονται στην δομή **struct fatten_operations fat12/16/32_ops** του αρχείου **fs/fat/fatent.c**. Οι λειτουργίες αρχείου ορίζονται στην δομή **struct file_operations fat_file_operations** του αρχείου **fs/fat/file.c**. Μερικές λειτουργίες inode ορίζονται στην δομή **struct inode_operations fat_file_inode_operations** του αρχείου **fs/fat/file.c**. Οι λειτουργίες των καταλόγων ορίζονται στην δομή **struct inode_operations msdos_dir_inode_operations** του αρχείου **fs/fat/namei_msdos.c** για το FAT και **fs/fat/namei_vfat.c** για το VFAT.

4. Προετοιμασία

Κατεβάστε την εικονική μηχανή MYY601Lab2.zip (4.5GB) και αποσυμπιέστε την σε τοπικό δίσκο (τουλάχιστο 8.5GB). Η εικονική μηχανή τρέχει την έκδοση 10 (“Buster”) του Debian Linux 64.bit. Κατεβάστε και εγκαταστήστε τον VMware Player v15 στο μηχάνημά σας. Η μηχανή είναι ρυθμισμένη να χρησιμοποιεί 1 πυρήνα και 1GB RAM, αλλά μπορείτε να την αλλάξετε από τις ρυθμίσεις της εικονικής μηχανής. Για να εκμεταλλευτείτε το παραθυρικό περιβάλλον, εισάγετε κατάσταση πλήρους οθόνης πιέζοντας το κατάλληλο εικονίδιο πάνω αριστερά του VMware. Μπορείτε να εισέλθετε ως απλός χρήστης με το όνομα myy601 και το ίδιο ως κωδικό ασφάλειας. Αν χρειαστεί να εγκαταστήσετε επιπλέον πακέτα στο σύστημα Linux, μπορείτε να εισέλθετε ως χρήστης root (με κωδικό myy601) και να εκτελέσετε **apt install <package>**. Η εικονική μηχανή έχει εγκατεστημένα τα εργαλεία ανάπτυξης C/C++. Για διευκόλυνση μπορείτε να κάνετε αντιγραφή copy-paste κειμένου και αρχείων μεταξύ του λειτουργικού συστήματος εντός της εικονικής μηχανής και του λειτουργικού συστήματος στο οποίο εκτελείται η εικονική μηχανή, για παράδειγμα για να παραδώσετε τον κώδικα που έχετε γράψει για την άσκηση.

Μπορείτε να περιηγηθείτε στον κώδικα της LKL επισκεπτόμενοι τον σύνδεσμο LKL DOX με ένα φυλλομετρητή. Επίσης μπορείτε να κάνετε αναζήτηση για μια συμβολοσειρά σε έναν κατάλογο αναδρομικά χρησιμοποιώντας την εντολή **~/bin/search <string>**. Ανοίξτε ένα τερματικό στην εικονική μηχανή και μετακινηθείτε στον κατάλογο **lkl/lkl-source** που περιέχει τον πηγαίο κώδικα της LKL καθώς και σχετικές εφαρμογές. Μεταγλωττίστε την βιβλιοθήκη με την εντολή **make -j 4 -C tools/lkl**. Στην συνέχεια, μπορείτε να καθαρίσετε την τρέχουσα μεταγλώττιση με **make -C tools/lkl clean** στον κατάλογο **lkl/lkl-source**. Εάν ρυθμίσετε την εικονική μηχανή να χρησιμοποιεί περισσότερους υπολογιστικούς πυρήνες τροποποιήστε ανάλογα την παράμετρο -j της εντολής make για περισσότερο παραλληλισμό.

Μπορείτε να μεταγλωττίσετε δοκιμαστικές εφαρμογές με την εντολή **make test** στον κατάλογο **tools/lkl** και αν χρειαστεί μπορείτε να καθαρίσετε την τρέχουσα μεταγλώττιση με **make clean** στον κατάλογο **tools/lkl**. Με **make test** στο **tools/lkl** καλείτε το **tests/boot.sh** που δημιουργεί ένα τοπικό αρχείο **/tmp/vfatfile** και το προσθέτει ως δίσκο στην εφαρμογή **tests/boot** πριν το κάνει mount. Στην συνέχεια καλεί μια σειρά από ελέγχους για να επιβεβαιώσει την σωστή λειτουργία της LKL.

Η μορφοποίηση του αρχείου δίσκου γίνεται με το εργαλείο **/sbin/mkfs.vfat** του συστήματος αρχείων VFAT. Επίσης μπορείτε να καλέσετε απευθείας την **boot** με **tests/boot -t vfat -d /tmp/vfatfile -p -P 0**.

Εναλλακτικά μπορείτε να προσπελάσετε το αρχείο απευθείας από το λειτουργικό σύστημα της εικονικής μηχανής με την εντολή **mount -t vfat -o loop /tmp/vfatfile /vfat** ως χρήστης root (**login root**). Μετά μπορείτε να μετακινηθείτε στον κατάλογο **/vfat** και να προσπελάσετε το σύστημα αρχείων σαν να ήταν αποθηκευμένο σε διαμέριση του δίσκου αντί σε αρχείο. Αν τροποποιήσετε το σύστημα **/vfat** μέσω της LKL μπορεί να χρειαστεί να κάνετε **umount /vfat** ως root και να ξανακάνετε **mount** για να δείτε τις τελευταίες αλλαγές.

Στον κατάλογο **lkl/lkl-source/tools/lkl** μπορείτε να βρείτε τις εφαρμογές **cptofs** και **cpfromfs** που επιτρέπουν την μετακίνηση των αρχείων μεταξύ του συστήματος αρχείων της εικονικής μηχανής και του συστήματος αρχείων της LKL. Αυτό αφορά ξεχωριστά αρχεία αλλά και καταλόγους αναδρομικά. Για παράδειγμα, η εντολή ., **./cptofs -i /tmp/vfatfile -p -t vfat lklfuse.c /** θα αντιγράψει το αρχείο **lklfuse.c** στον κατάλογο **/** του **/tmp/vfatfile**.

5. Άσκηση

Σας ζητείται να υλοποιήσετε αρχείο καταγραφής τροποποιήσεων για το σύστημα αρχείων FAT στην Linux Kernel Library. Για να διαχειριστείτε την πολυπλοκότητα ακολουθείστε τα ακόλουθα βήματα:

i. Για να κατανοήσετε την λειτουργία του συστήματος αρχείων FAT μπορείτε να χρησιμοποιήσετε τις εφαρμογές boot και cptofs ώστε να δείτε τις λειτουργίες που εμπλέκονται κατά την προσπέλαση του αρχείου **/tmp/vfatfile**. Μπορείτε να εκτελέσετε τις εφαρμογές με τον αποσφαλματωτή **gdb** προκειμένου να προχωρήσετε βηματικά στις συναρτήσεις που καλούνται κατά την εκτέλεση. Επιπλέον, μπορείτε να τυπώσετε μηνύματα στην κονσόλα προσθέτοντας την συνάρτηση **printk(KERN_INFO "...", ...)** στον πηγαίο κώδικα της LKL (π.χ., στον κατάλογο **fat/** ή **mm/**) και μεταγλωττίζοντας ξανά την βιβλιοθήκη και την εφαρμογή (μπορεί να χρειαστεί clean για να γίνουν ορατές οι αλλαγές στο εκτελέσιμο).

ii. Στην συνέχεια, μπορείτε να αποθηκεύσετε εγγραφές των τροποποιήσεων του FAT κατά την εκτέλεση των εφαρμογών στο τοπικό σύστημα αρχείων της εικονικής μηχανής χρησιμοποιώντας κανονικές κλήσεις συστήματος εισόδου/εξόδου (π.χ., **open**, **write**, κλπ). Σας ζητείται να καταγράψετε αλλαγές των δομών του συστήματος FAT όπως superblock, file allocation table, directory entries και δεδομένα αρχείων όχι τις κλήσεις συστήματος.

iii. Μια πιο ρεαλιστική υλοποίηση της καταγραφής θα χρησιμοποιήσει τις κλήσεις του Linux (π.χ., ., **sys_open**, **sys_write**, κλπ) σε αρχείο που είναι αποθηκευμένο στο ίδιο το σύστημα FAT που παρακολουθείτε. Αναμένεται να καλέσετε τις συναρτήσεις αυτές από τον πηγαίο κώδικα της LKL σε θέσεις όπου πραγματοποιούνται οι τροποποιήσεις των δομών. Οι εγγραφές θα πρέπει να αντιστοιχούν σε εσωτερικές δομές του FAT όπως είναι το superblock, το file allocation table, τα directory entries και τα δεδομένα των αρχείων.

iv. Είστε υπεύθυνοι να υλοποιήσετε ελέγχους που δείχνουν ότι οι εγγραφές αντιστοιχούν στις αναμενόμενες τροποποιήσεις από διάφορες εκτελέσεις εφαρμογών που προκαλούν δημιουργία αρχείων και καταλόγων στο σύστημα αρχείων FAT (π.χ., χρησιμοποιώντας **tests/boot** ή **cptofs**).

v. **Bonus 20%**. Δημιουργήσετε ένα αρχείο καταγραφής σε σύστημα αρχείων FAT που αρχικά είναι άδειο και στην συνέχεια εκτελούνται εφαρμογές που το τροποποιούν. Η επιπλέον υλοποίησή σας αφορά την ανακατασκευή σε άλλο αρχικά άδειο σύστημα αρχείων με την επανάληψη (replay) των τροποποιήσεων που περιέχονται στο αρχείο καταγραφής. Η αναδημιουργία του συστήματος αρχείων θα πρέπει να βασίζεται στην καταγραφή των αλλαγών που αφορούν εσωτερικές δομές του συστήματος αρχείων FAT και όχι απλή λίστα των κλήσεων συστήματος της εφαρμογής.

Θα πρέπει να ετοιμάσετε αναφορά που τεκμηριώνει τα βήματα υλοποίησης που ακολουθήσατε, όλες τις γραμμές πηγαίου κώδικα που προσθέσατε ή τροποποιήσατε και το αποτέλεσμα των ελέγχων που ετοιμάσατε και τρέξατε. Θα πρέπει να τεκμηριώσετε όλα τα βήματα συμπεριλαμβανομένων των κλήσεων printk που προσθέσατε για να κατανοήσετε την λειτουργία του FAT. Επιπλέον, θα πρέπει να προσθέσετε σχόλια στον κώδικα που τροποποιήσατε για να δείξετε το αναμενόμενο αποτέλεσμα των γραμμών που προσθέσατε ή αλλάξατε.

6. Τι θα παραδώσετε

Μπορείτε να προετοιμάσετε την λύση ατομικά ή σε ομάδες των δύο ατόμων. Η υποβολή θα γίνει από ένα μέλος της ομάδας μόνο. Υποβολή μετά την λήξη της προθεσμίας οδηγεί σε αφαίρεση 10% του βαθμού ανά ημέρα μέχρι 50%. Για παράδειγμα, αν στείλετε την λύση σας 1 ώρα μετά την προθεσμία, ο μέγιστος βαθμός που μπορείτε να λάβετε είναι 9 στα 10. Αν στείλετε την λύση μία εβδομάδα μετά την προθεσμία, ο μέγιστος βαθμός σας πέφτει σε 5 από τα 10. Υποβάλετε την λύση σας εγκαίρως με την εντολή

/usr/local/bin/turnin lab2_21@myy601 Report.pdf lkl-source.zip

Θα βαθμολογηθείτε σύμφωνα με την περιγραφή που παραδίδετε στο **Report.pdf** και την υλοποίηση που περιέχεται στο **lkl-source.zip**. Στο αρχείο **lkl-source.zip** θα πρέπει να συμπεριλάβετε μόνο τον πηγαίο κώδικα που έχετε τροποποιήσει καθώς και νέα αρχεία που δημιουργήσατε για την υλοποίηση ή τον έλεγχο. Ο πηγαίος κώδικας και οι έλεγχοι θα πρέπει να μεταγλωττίζονται και να εκτελούνται στην εικονική μηχανή που σας δίνεται με την εκφώνηση.