ΕΘΝΙΚΟ ΜΕΤΣΟΒΙΟ ΠΟΛΥΤΕΧΝΕΙΟ
ΣΧΟΛΗ ΗΛΕΚΤΡΟΛΟΓΩΝ ΜΗΧΑΝΙΚΩΝ ΚΑΙ ΜΗΧΑΝΙΚΩΝ ΥΠΟΛΟΓΙΣΤΩΝ
ΤΟΜΕΑΣ ΤΕΧΝΟΛΟΓΙΑΣ ΠΛΗΡΟΦΟΡΙΚΗΣ ΚΑΙ ΥΠΟΛΟΓΙΣΤΩΝ
ΕΡΓΑΣΤΗΡΙΟ ΥΠΟΛΟΓΙΣΤΙΚΩΝ ΣΥΣΤΗΜΑΤΩΝ
http://www.cslab.ece.ntua.gr

Εργαστήριο Λειτουργικών Συστημάτων

7ο εξάμηνο, Ακαδημαϊκή περίοδος 2023–2024

Άσκηση 3
Συστήματα Αρχείων σε Περιβάλλον Linux

Εργαστήριο Υπολογιστικών Συστημάτων Ε.Μ.Π.
os-lab@lists.cslab.ece.ntua.gr

Δεκέμβριος 2023

1 Εισαγωγή

Αντικείμενο της παρούσας άσκησης είναι η εξοικείωσή σας με τα συστήματα αρχείων που χρησιμοποιούνται για την οργάνωση των δεδομένων (αρχείων, καταλόγων, κλπ) σε συσκευές μπλοκ (σκληροί δίσκοι, USB flash drives, CD-ROM, δισκέτες κλπ). Η άσκηση χωρίζεται σε δύο μέρη. Στο πρώτο μέρος σκοπός είναι να εξοικειωθείτε με την οργάνωση των περιεχομένων ενός δίσκου που περιέχει ένα σύστημα αρχείων ext2, χρησιμοποιώντας εργαλεία του χώρου χρήστη όπως τα dumpe2fs, hexdump, κλπ. Στο δεύτερο μέρος της άσκησης σκοπός είναι να εξοικειωθείτε με την υλοποίηση ενός συστήματος αρχείων στον πυρήνα του Linux. Πιο συγκεκριμένα, σας δίνουμε το μεγαλύτερο μέρος της υλοποίησης μίας πολύ απλουστευμένης έκδοχής του ext2, την οποία ονομάζουμε ext2-lite, και σας ζητάμε να συμπληρώσετε μικρά και απλά κομμάτια κώδικα ώστε να ολοκληρώσετε την υλοποίηση του συστήματος αρχείων.

2 $1^{ο}$ Μέρος - Το σύστημα αρχείων ext2

Στο πρώτο μέρος της άσκησης σας δίνουμε εικόνες δίσκων (virtual disk images) και σας ζητάμε να απαντήσετε μια σειρά από ερωτήσεις/προκλήσεις σε σχέση με τα συστήματα αρχείων που περιέχουν αυτές οι εικόνες. Για να το πετύχετε, πρέπει να καταλάβετε τη δομή ενός συστήματος αρχείων και να είστε σε θέση να χρησιμοποιήσετε μια σειρά από εργαλεία και τεχνικές ώστε να εξερευνήσετε τις εικόνες που σας δίνουμε.

2.1 Δύο προσεγγίσεις

Στα ακόλουθα υποθέτουμε ότι κάθε φορά συνδέετε τις εικόνες δίσκων που σας δίνουμε ως εικονικούς δίσκους στην εικονική μηχανή QEMU-KVM `utopia` που ήδη έχετε από την προηγούμενη άσκηση. Για το σκοπό αυτό, πρέπει να επεκτείνετε το script `utopia.sh` που ήδη χρησιμοποιείτε ώστε η εικονική μηχανή να βλέπει την κάθε εικόνα ως έναν επιπλέον εικονικό δίσκο.

Έχοντας ολοκληρώσει αυτό το βήμα, υπάρχουν δύο βασικές προσεγγίσεις για να λύσετε τις προκλήσεις:

tools: Προσάρτηση συστήματος αρχείων στο Linux και χρήση έτοιμων εργαλείων
: Μπορείτε να προσαρτήσετε το σύστημα αρχείων σε κατάλογο του συστήματος [π.χ. στον `/mnt`] με την εντολή `mount` και να χρησιμοποιήσετε όλα τα εργαλεία που ήδη παρέχει το Linux ώστε να εξερευνήσετε το περιεχόμενό

της, ενδεικτικά `ls`, `cat`, `stat`, `df`, `du`, και πολλά άλλα, δείτε και την §2.2. Επιπλέον, μπορείτε να χρησιμοποιήσετε μια σειρά έτοιμων εργαλείων που χειρίζονται απευθείας το σύστημα αρχείων, όπως `tune2fs`, `dumpe2fs`, `debugfs`.

hexedit: Απευθείας εξερεύνηση των εικόνων με χρήση hex editor Στην περίπτωση αυτή μπορείτε να χρησιμοποιήσετε μόνο έναν hex viewer/editor, όπως τα `hexdump`, `xxd`, `hexedit`, ώστε να μελετήσετε byte προς byte το περιεχόμενο του εικονικού δίσκου, π.χ. `/dev/vdb`, και να εξάγετε στοιχεία για το περιεχόμενό του συστήματος αρχείων με απευθείας πρόσβαση στις δομές του.

Για κάθε πρόκληση, σας ζητάμε να περιγράψετε βήμα προς βήμα τον τρόπο επίλυσής της *και με τους δύο τρόπους*.

2.2 Χρήσιμα Εργαλεία

Τα παρακάτω εργαλεία θα σας φανούν πολύ χρήσιμα για να λύσετε τις προκλήσεις:

Για κάθε ένα από αυτά μπορείτε να βρείτε πολλές χρήσιμες πληροφορίες στο manpage του, και στο δίκτυο: `mount`, `fdisk`, `lsblk`, `file`, `df`, `du`, `stat`, `blkid`, `tune2fs`, `dumpe2fs`, `debugfs`, `hexdump`, `hexedit`, `xxd`.

2.3 Η εικόνα *fsdisk1.img*

Απαντήστε τις παρακάτω ερωτήσεις/προκλήσεις. Μερικές από αυτές αναφέρονται γενικά στο σύστημα αρχείων ext2, ενώ άλλες ζητούν στοιχεία για το συγκεκριμένο σύστημα αρχείων που περιέχει η εικόνα `fsdisk1.img` που σας δίνεται.

Για κάθε πρόκληση που ζητά γενικές πληροφορίες για το σύστημα αρχείων ext2, δικαιολογήστε την απάντησή σας αναφέροντας τις πηγές σας. Γίνετε όσο πιο συγκεκριμένοι μπορείτε.

Για κάθε πρόκληση που ζητά δεδομένα από την εικόνα `fsdisk1.img` περιγράψτε τη λύση της πρόκλησης *και με τις δύο* προσεγγίσεις, **mount** και **hexedit**, δείχνοντας αναλυτικά τις εντολές που χρειάστηκε να εκτελέσετε, εκτός αν η πρόκληση υποδεικνύει κάτι διαφορετικό.

1. Τροποποιήστε κατάλληλα το αρχείο `utopia.sh` ώστε να προσθέσετε στην εικονική μηχανή utopia έναν επιπλέον δίσκο για την εικόνα `fsdisk1.img`. Ποια είναι η προσθήκη που κάνατε; Ποια συσκευή στο utopia είναι αυτή που μόλις προσθέσατε;

2. Τι μέγεθος έχει ο δίσκος που προσθέσατε στο utopia;

3. Τι σύστημα αρχείων περιέχει;
4. Πότε *ακριβώς* δημιουργήθηκε αυτό το σύστημα αρχείων; Δείξτε τη χρονοσφραγίδα [timestamp].
5. Πότε *ακριβώς* προσαρτήθηκε τελευταία φορά; Δείξτε τη χρονοσφραγίδα.
6. Σε ποιο μονοπάτι προσαρτήθηκε τελευταία φορά;
7. Πότε *ακριβώς* τροποποιήθηκε τελευταία φορά; Δείξτε τη χρονοσφραγίδα.
8. Τι είναι το μπλοκ σε ένα σύστημα αρχείων;
9. Τι μέγεθος μπλοκ [block size] χρησιμοποιεί αυτό το σύστημα αρχείων;
10. Τι είναι το inode σε ένα σύστημα αρχείων;
11. Τι μέγεθος έχει το inode σε αυτό το σύστημα αρχείων;
12. Πόσα διαθέσιμα μπλοκ και πόσα διαθέσιμα inodes υπάρχουν σε αυτό το σύστημα αρχείων;
13. Τι είναι το superblock στο σύστημα αρχείων ext2;
14. Πού βρίσκεται μέσα στον δίσκο σε ένα σύστημα αρχείων ext2;
15. Για ποιο λόγο έχει νόημα να υπάρχουν εφεδρικά αντίγραφα του superblock στο σύστημα αρχείων ext2;
16. Σε ποια μπλοκ βρίσκονται αποθηκευμένα εφεδρικά αντίγραφα του superblock σε αυτό το σύστημα αρχείων;
17. Τι είναι ένα block group στο σύστημα αρχείων ext2;
18. Πόσα block groups έχει ένα σύστημα αρχείων ext2 και πώς κατανέμονται;
19. Πόσα block groups περιέχει αυτό το σύστημα αρχείων;
20. Τι είναι ο block group descriptor στο σύστημα αρχείων ext2;
21. Για ποιο λόγο έχει νόημα να υπάρχουν εφεδρικά αντίγραφα των block group descriptors στο σύστημα αρχείων ext2;
22. Σε ποια μπλοκ βρίσκονται αποθηκευμένα εφεδρικά αντίγραφα των block group descriptors σε αυτό το σύστημα αρχείων;
23. Τι είναι το block bitmap και τι το inode bitmap; Πού βρίσκονται μέσα στον δίσκο;

24. Τι είναι τα inode tables; Πού βρίσκονται μέσα στον δίσκο;

25. Τι πεδία περιέχει το κάθε inode; Πού αποθηκεύεται μέσα στον δίσκο;

26. Πόσα μπλοκ και πόσα inodes περιέχει το κάθε block group σε αυτό το σύστημα αρχείων;

27. Σε ποιο inode αντιστοιχεί το αρχείο `/dir2/helloworld` σε αυτό το σύστημα αρχείων;

28. Σε ποιο block group αντιστοιχεί αυτό το inode;

29. Σε ποιο μπλοκ του δίσκου υπάρχει το inode table που περιέχει το παραπάνω inode;

30. Δείξτε όλα τα πεδία αυτού του inode [128 bytes]

31. Σε ποιο μπλοκ είναι αποθηκευμένα τα δεδομένα αυτού του αρχείου;

32. Τι μέγεθος έχει αυτό το αρχείο;

33. Δείξτε τα περιεχόμενα αυτού του αρχείου.

2.4 Η εικόνα *fsdisk2.img*

Σας δίνεται μια νέα εικόνα, `fdisk2.img`. Απαντήστε τις παρακάτω προκλήσεις/ερωτήσεις ακριβώς όπως και για την `fsdisk1.img`. Θυμηθείτε να περιγράψετε τη λύση κάθε πρόκλησης *και με τις δύο* προσεγγίσεις, **mount** και **hexedit**, εκτός αν η πρόκληση υποδεικνύει κάτι διαφορετικό.

1. Συνδέστε την εικόνα του δίσκου στην εικονική μηχανή σας, όπως κάνατε και για την εικόνα `fsdisk1.img` και προσαρτήστε τη στον κατάλογο `/mnt`.

2. Χρησιμοποιήστε την εντολή `touch` για να δημιουργήσετε ένα νέο κενό αρχείο `/file1` μέσα στο συγκεκριμένο σύστημα αρχείων. Βεβαιωθείτε ότι η εντολή σας αναφέρεται πράγματι στο συγκεκριμένο σύστημα αρχείων [σε ποιον κατάλογο το έχετε προσαρτήσει;], κι όχι στον ριζικό κατάλογο του συστήματος.

3. Πέτυχε η εντολή; Αν όχι, τι πρόβλημα υπήρξε;

4. Ποια κλήση συστήματος προσπάθησε να τρέξει η `touch`, και με ποιον κωδικό λάθους απέτυχε; Υποστηρίξτε την απάντησή σας με χρήση της εντολής `strace`.

5. Πόσα αρχεία και πόσους καταλόγους περιέχει το συγκεκριμένο σύστημα αρχείων;

6. Πόσο χώρο καταλαμβάνουν τα δεδομένα και τα μεταδεδομένα του συγκεκριμένου συστήματος αρχείων;

7. Πόσο είναι το μέγεθος του συγκεκριμένου συστήματος αρχείων;

8. Πόσα μπλοκ είναι διαθέσιμα/ελεύθερα στο συγκεκριμένο σύστημα αρχείων; Ισοδύναμα, έχει ελεύθερο χώρο το συγκεκριμένο σύστημα αρχείων;

9. Αφού υπάρχουν διαθέσιμα μπλοκ, τι σας αποτρέπει από το να δημιουργήσετε νέο αρχείο;

2.5 Η εικόνα *fsdisk3.img*

Η εικόνα `fsdisk3.img` περιέχει ένα σύστημα αρχείων ext2 *με λάθη*. Είναι ένα σύστημα αρχείων που έχει πάθει μία ή περισσότερες μορφές αλλοίωσης δεδομένων [corruption]. Στις ακόλουθες προκλήσεις καλείστε να εντοπίσετε και να επιδιορθώσετε τις αλλοιώσεις αυτές.

ΠΡΟΣΟΧΗ: Για να απαντήσετε τις προκλήσεις αυτές, θα χρειαστεί να αλλάξετε το περιεχόμενο της εικόνας. Θυμηθείτε να τηρείτε αντίγραφα ασφαλείας, και να ξεκινήσετε από την καθαρή εικόνα όπως σας δίνεται για την τελική εκτέλεση της άσκησης. Επιβεβαιώστε το περιεχόμενο της εικόνας με `sha256sum`, και συγκρίνετε με το hash που περιέχει το όνομά της.

1. Ποιο εργαλείο στο Linux αναλαμβάνει τον έλεγχο ενός συστήματος αρχείων ext2 για αλλοιώσεις;

2. Ποιοι παράγοντες θα μπορούσαν δυνητικά να οδηγήσουν σε αλλοιώσεις στο σύστημα αρχείων; Αναφέρετε ενδεικτικά δέκα πιθανές αλλοιώσεις.

3. Τρέξτε το εργαλείο αυτό και επιδιορθώστε το σύστημα αρχείων. Αναφέρετε *όλες* τις αλλοιώσεις που εντοπίσατε, εξαντλητικά.

4. Επαναφέρετε το δίσκο στην πρότερή του κατάσταση, από την αρχική εικόνα. Εντοπίστε τις αλλοιώσεις με χρήση της μεθόδου **hexedit**.

5. Επιδιορθώστε κάθε αλλοίωση ξεχωριστά με χρήση της μεθόδου **hexedit**. Για κάθε μία από τις αλλοιώσεις που επιδιορθώνετε, τρέξτε το εργαλείο `fsck` *με τρόπο που δεν προκαλεί καμία αλλαγή* [“dry run”] και επιβεβαιώστε ότι πλέον δεν την εντοπίζει.

2.6 Πληροφορίες για το σύστημα αρχείων ext2

Στα πλαίσια της παρούσας εργασίας καλείστε να αναζητήσετε πληροφορίες σχετικά με το σύστημα αρχείων ext2 και να κατανοήσετε τον τρόπο με τον οποίο οργανώνονται τα δεδομένα σε έναν δίσκο που περιέχει ένα σύστημα αρχείων ext2. Θα σας φανούν πολύ χρήσιμες οι παρακάτω πληροφορίες για να καταλάβετε την οργάνωση του ext2:

- https://en.wikipedia.org/wiki/Ext2
- https://wiki.osdev.org/Ext2
- https://www.nongnu.org/ext2-doc/
- http://www.science.smith.edu/~nhowe/262/oldlabs/ext2.html
- https://docs.kernel.org/filesystems/ext2.html

3 $2^{ο}$ Μέρος - Το σύστημα αρχείων ext2-lite

Στο δεύτερο μέρος της άσκησης θα ασχοληθείτε με τα συστήματα αρχείων απο τη μεριά του πυρήνα του Linux. Πιο συγκεκριμένα, σας δίνουμε κώδικα σε C, βασισμένο στον αρχικό κώδικα του ext2 που μπορείτε να βρείτε και στον πηγαίο κώδικα του πυρήνα[1], ο οποίος υλοποιεί ένα αρκετά απλουστευμένο ext2 σύστημα αρχείων το οποίο θα ονομάσουμε ext2-lite. Απο εσάς ζητάμε να συμπληρώσετε μικρά κομμάτια του κώδικα ώστε να ολοκληρώσετε την υλοποίηση. Στα κομμάτια που περιμένουμε να συμπληρώσετε εσείς έχουμε βάλει σχόλια της μορφής `/* ? */`. Πιο συγκεκριμένα, τα κομμάτια που θέλουμε να συμπληρώσετε είναι:

1. Τις συναρτήσεις `init_ext2_fs` και `exit_ext2_fs` στο αρχείο `super.c`. Εδώ θα πρέπει να συμπληρώσετε τις κατάλληλες γραμμές κώδικα ώστε να γίνεται σωστά η καταχώρηση και διαγραφή αντίστοιχα του ext2-lite συστήματος αρχείων στον πυρήνα του Linux. Μπορείτε να βεβαιωθείτε οτι το έχετε κάνει σωστά εκτελώντας την εντολή `cat /proc/filesystems` αφού εισάγετε το module στον πυρήνα και ελέγχοντας οτι υπάρχει η εγγραφή για το ext2-lite.

2. Τη συνάρτηση `ext2_find_entry` στο αρχείο `dir.c` η οποία θα πρέπει να σκανάρει τη λίστα με τα directory entries του. Συναρτήσεις που θα σας φανούν χρήσιμες εδώ είναι:

[1] https://elixir.bootlin.com/linux/v5.10/source/fs/ext2

- `struct page *ext2_get_page(struct inode *dir, unsigned long n, int quiet)`: Χρησιμοποιείται για την ανάγνωση σελίδων ενός καταλόγου, για αυτό και η πρώτη παράμετρος ονομάζεται dir. Η συνάρτηση επιστρέφει την n-οστή σελίδα του inode dir. Η παράμετρος quiet αφορά διαγνωστικά μηνύματα της ext2_check_page, οπότε δεν επηρεάζει την ορθότητα της υλοποίησης σας. Μπορείτε να ελέγξετε για σφάλματα κατα την επιστροφή της συνάρτησης με την μακροεντολή `IS_ERR(page)` και να επιστρέψετε κατάλληλη τιμή σφάλματος κι εσείς με την `ERR_CAST(page)`.
- `void *page_address(const struct page *page)`: Επιστρέφει την διεύθυνση του χώρου διευθύνσεων του πυρήνα, η οποία αντιστοιχεί στην δοθείσα σελίδα.
- `unsigned ext2_last_byte(struct inode *inode, unsigned long page_nr)`: Επιστρέφει το τελευταίο χρησιμοποιούμενο byte της σελίδας `page_nr` του inode. Χρησιμοποιείται για σελίδες που περιέχουν directory entries του ext2 (`ext2_dirent`), ώστε να γνωρίζουμε μέχρι ποιο σημείο να ψάξουμε.
- `ext2_dirent *ext2_next_entry(ext2_dirent *p)`: Επιστρέφει το επόμενο directory entry απο το `p`.
- `void ext2_put_page(struct page *page)`: Η έκφραση put εδώ έχει την έννοια του release. Οφείλει να συνοδεύει κάθε κλήση της `ext2_get_page`, όταν πλέον δεν χρειαζόμαστε άλλο την σελίδα. Μειώνει μεταξύ άλλων έναν reference counter, και βοηθάει το memory management να γνωρίζει πως να διαχειριστεί την σελίδα αναλόγως αν υπάρχει ακόμα κάποιος που την χρησιμοποιεί ή όχι. Θα πρέπει πριν επιστρέψει η `ext2_find_entry` να έχει γίνει release κάθε σελίδα στην οποία έγινε πρόσβαση.
- `int ext2_match(int len, const char *const name, ext2_dirent *de)`: Συγκρίνει το `de->name` με το `name` και επιστρέφει 1 αν ταιριάζουν αλλιώς 0.

3. Τη συνάρτηση `ext2_get_inode` στο αρχείο `inode.c`. Η συνάρτηση αυτή εντοπίζει στο δίσκο ένα ext2 inode με βάση τον αριθμό του. Αρχικά πρέπει απ'τον αριθμό του inode να βρείτε σε ποιο block group ανήκει και να βρείτε το κατάλληλο group descriptor (`ext2_group_desc`). Στη συνέχεια να διαβάσετε το block που περιέχει το inode table για το συγκεκριμένο block group και αφού υπολογίσετε το offset του inode που ψάχνετε μέσα στο inode table (πάλι με βάση τον αριθμό του inode), να επιστρέψετε δείκτη στο κατάλληλο σημείο του buffer που αντιστοιχεί στο inode table. Συναρτήσεις που θα σας φανούν χρήσιμες εδώ είναι:

- `struct ext2_group_desc *ext2_get_group_desc(struct super_block *sb, unsigned int block_group, struct buffer_head **bh)`: Εντοπίζει και επιστρέφει τον group descriptor για το ζητούμενο block group. Στην παράμετρο `bh` επιστρέφει δείκτη στον buffer που αποθηκεύει το συγκεκριμένο block group descriptor. Επειδή δε μας ενδιαφέρει να τροποιήσουμε τον συγκεκριμένο descriptor, μπορούμε να θέσουμε αυτή την παράμετρο σε `NULL`.
- Η μακροεντολή `le32_to_cpu` του πυρήνα μπορεί να χρησιμοποιηθεί για την μετατροπή little endian πεδίων του ext2 στη μορφή που υποστηρίζει ο επεξεργαστής του μηχανήματος σας.
- `struct buffer_head *sb_bread(struct super_block *sb, sector_t block)`: Διαβάζει ένα block απο το δίσκο και επιστρέφει έναν δείκτη στη δομή `buffer_head` που συνδέεται με αυτό το block. Δε χρειάζεται να έχετε λεπτομερή γνώση του τι είναι ένα `buffer_head`, αρκεί να γνωρίζετε οτι μπορείτε να αποκτήσετε πρόσβαση στα δεδομένα του buffer μέσω του πεδίου `bh->b_data`.

4. Τη συνάρτηση `ext2_iget` στο αρχείο `inode.c`. Η συγκεκριμένη συνάρτηση εντοπίζει ένα inode στο σύστημα αρχείων δεδομένου του αριθμού του inode (καλώντας την `ext2_get_inode` που συμπληρώσατε προηγουμένως). Δημιουργεί ένα VFS inode και το αρχικοποιεί με τα κατάλληλα δεδομένα τα οποία διαβάζει απο το ext2 inode που διαβάζει απο το δίσκο. Εσείς εδώ πρέπει απλά να συμπληρώσετε κατάλληλα την αρχικοποίηση των πεδίων του VFS inode που περιέχουν τους δείκτες στις μεθόδους/συναρτήσεις του ext2-lite, δηλαδή τα πεδία `inode->i_op`, `inode->i_fop` και `inode->i_mapping->a_ops`.

5. Τη συνάρτηση `ext2_allocate_in_bg` στο αρχείο `balloc.c`. Αυτή η συνάρτηση εντοπίζει το πρώτο διαθέσιμο block μέσα στο δεδομένο group και ξεκινώντας απο αυτό δεσμεύει το πολύ `count` συνεχόμενα blocks. Σε περίπτωση που δεν υπάρχει κανένα διαθέσιμο block επιστρέφει -1, σε αντίθετη περίπτωση επιστρέφει το offset μέσα στο block bitmap του πρώτου block που δέσμευσε. Επίσης τροποποιεί κατάλληλα την παράμετρο `count` ανάλογα με το πόσα συνεχόμενα blocks κατάφερε να δεσμεύσει. Συναρτήσεις που θα σας φανούν χρήσιμες εδώ είναι:

   - `unsigned long find_next_zero_bit_le(const void *addr, unsigned long size, unsigned long offset)`: Επιστρέφει την θέση του πρώτου μηδενικού bit, ξεκινώντας από την διεύθυνση addr με πιθανό offset, ελέγχοντας το πολύ size σε πλήθος bits. Αν αποτύχει επιστρέφει θέση μεγαλύτερη του τελευταίου bit που θέλουμε να εξετάσει. Λαμβάνουμε

υπόψη ότι τα δεδομένα του buffer head είναι αποθηκευμένα σε little endian, για αυτό και το πρόθεμα `_le` της συνάρτησης. Για μας, η διεύθυνση θα αντιστοιχεί στο πεδίο `b_data` ενός `buffer_head` που δείχνει σε bitmap των μεταδεδομένων του συστήματος αρχείων.

- `ext2_{set,clear}_bit_atomic(l, nr, addr)`: Θέτουμε ή μηδενίζουμε το bit nr της διεύθυνσης addr, υπό το κλείδωμα l. Κατάλληλο για ενημέρωση των bitmaps του συστήματος αρχείων.

Αφού ολοκληρώσετε τον κώδικα θα πρέπει να μπορείτε να καταχωρήσετε το ext2-lite στον πυρήνα του linux, εισάγοντας το module σας. Επίσης, πρέπει να μπορείτε να προσαρτήσετε κάποιο δίσκο με ext2-lite και να εκτελέσετε τις βασικές λειτουργίες που αφορούν συστήματα αρχείων στο Linux, όπως, open, read, write, rm, mkdir, rmdir, κλπ. Μπορείτε να ελέγξετε την ορθή λειτουργία του module σας είτε χρησιμοποιώντας εντολές στο τερματικό (π.χ., cat, echo, rm, touch, ...) είτε γράφοντας κάποιο δικό σας πρόγραμμα ελέγχου που εκτελεί τις κατάλληλες κλήσεις συστημάτων.

4 Εξέταση άσκησης και αναφορά

Η προθεσμία για την εξέταση της άσκησης θα ανακοινωθεί στη σελίδα του μαθήματος.

Μετά την εξέταση κάθε ομάδα θα πρέπει να συντάξει σύντομη αναφορά, η οποία θα περιγράφει τον τρόπο με τον οποίο απαντήσατε/λύσατε κάθε μία από τις προκλήσεις. Θυμηθείτε ότι, όσο αφορά το πρώτο κομμάτι της άσκησης, πρέπει να συμπεριλάβετε λύση και με τη μέθοδο **tools** και με τη μέθοδο **hexedit**, εκτός αν η πρόκληση ζητάει ρητά κάτι διαφορετικό.