Τμήμα Ηλεκτρολόγων Μηχανικών & Μηχανικών Υπολογιστών

Λειτουργικά Συστήματα (ECE318)

Ακαδημαϊκό Έτος 2022-2023

3η Εργαστηριακή Άσκηση

Τελευταία Ενημέρωση: 3 Απριλίου 2023

Περιεχόμενα

1 Εισαγωγικά

Το υποσύστημα διαχείρισης μνήμης στο λειτουργικό σύστημα Linux είναι υπεύθυνο για την εξυπηρέτηση όλων των αιτήσεων που αφορούν σε δέσμευση μνήμης και προέρχονται από το πυρήνα λειτουργικό σύστημα (όχι από εφαρμογές χρήστη). Πρόκειται για το υποσύστημα πάνω στο οποίο βασίζεται η υλοποίηση των κλήσεων kmalloc και kfree (τα αντίστοιχα των malloc και free που είναι διαθέσιμα για χρήση από τον κώδικα του λειτουργικού συστήματος). Οι αιτήσεις για δέσμευση περιοχών μνήμης διαχωρίζονται σε αυτές με μέγεθος πολλαπλάσιο του μεγέθους της σελίδας μνήμης του λειτουργικού συστήματος και σε αυτές με μέγεθος μικρότερο από το μέγεθος μίας σελίδας. Για τη διαχείριση των αιτήσεων που συγκαταλέγονται στη δεύτερη κατηγορία, ο πυρήνας του λειτουργικού συστήματος Linux παρέχει τρεις διαφορετικούς διαχειριστές μνήμης (*memory allocators*): το διαχειριστή *SLAB*, το διαχειριστή *SLUB* και το διαχειριστή *SLOB*. Οι δύο πρώτοι είναι αρκετά πολύπλοκοι και προορίζονται για χρήση σε πραγματικά συστήματα, με μεγάλη ποσότητα φυσικής μνήμης. Έχουν σχεδιασθεί με κύριο στόχο να ελαχιστοποιήσουν τον εξωτερικό κατακερματισμό (*external fragmentation*) της φυσικής μνήμης και να επιτρέπουν αποδοτική χρήση της ελεύθερης μνήμης.

Σε αντίθεση με τους παραπάνω διαχειριστές μνήμης, ο διαχειριστής *SLOB* (Simple List Of Blocks) αποτελεί ένα διαχειριστή που επιτρέπει ταχεία και αποδοτική διαχείριση μνήμης, με μικρές απαιτήσεις σε πόρους για την εκτέλεσή του. Αυτά τα χαρακτηριστικά τον καθιστούν κατάλληλο για χρήση σε συστήματα με ελάχιστους πόρους, όπως είναι τα ενσωματωμένα συστήματα.

Το κύριο μειονέκτημα του διαχειριστή *SLOB* είναι ο μεγάλος βαθμός του εξωτερικού κατακερματισμού που δημιουργεί. Ο διαχειριστής SLOB χρησιμοποιεί τον αλγόριθμο *First-Fit* για τη δέσμευση μνήμης, ο οποίος διευκολύνει την ταχεία δέσμευση μνήμης, αυξάνοντας όμως παράλληλα τον εξωτερικό κατακερματισμό.

Σε αυτή την εργασία καλείστε να τροποποιήσετε τον διαχειριστή SLOB και να αντικαταστήσετε τον αλγόριθμο *First-Fit* με τον αλγόριθμο *Best-Fit*, προκειμένου να ελαχιστοποιηθεί ο εξωτερικός κατακερματισμός κατά τη δέσμευση μνήμης[1]. Επίσης, θα υλοποιήσετε τις κατάλληλες κλήσεις συστήματος προκειμένου να παρέχετε ένα ποσοτικό μέτρο εκτίμησης του εξωτερικού κατακερματισμού που έχει προκαλέσει ο διαχειριστής σε μία δεδομένη χρονική στιγμή.

[1] Ο αλγόριθμος Best-Fit είναι και αυτός που προτάθηκε και από τον Linus Torvalds για την μείωση του κατακερματισμού που εμφανίζει ο SLOB.

1.1 Προαπαιτούμενα

Πριν ξεκινήσετε την υλοποίηση θα πρέπει να έχετε διαβάσει το κεφάλαιο 12 από το βιβλίο ”Linux Kernel Development”. Επίσης, θα πρέπει να έχετε μελετήσει τον κώδικα του αρχείου *mm/slob.c* του πηγαίου κώδικα του πυρήνα του Linux. Διαβάστε την επικεφαλίδα του αρχείου και προσπαθήστε να κατανοήσετε τους αλγορίθμους που χρησιμοποιεί ο διαχειριστής για τη δέσμευση μνήμης. Πιο συγκεκριμένα, προσπαθήστε να κατανοήσετε ότι η συνάρτηση *slob_page_alloc()* υλοποιεί τον αλγόριθμο *First-Fit* για τη δέσμευση της απαιτούμενης μνήμης.

Για αυτή την εργασία θα ξεκινήσετε από ένα ”καθαρό” δέντρο κώδικα του πυρήνα του Linux. Για να γίνει αυτό εκτελέστε τις ακόλουθες εντολές:

```
cd /usr/src
sudo rm -rf linux-5.4.86-dev
cp -R linux-5.4.86-orig linux-5.4.86-dev
```

Στη συνέχεια ακολουθήστε τις οδηγίες μεταγλώττισης του πυρήνα και τη δημιουργία του *.config* αρχείου από την εργασία 1.

1.2 Προθεσμία και Τρόπος Παράδοσης

Η άσκηση θα πρέπει να παραδοθεί έως τα **μεσάνυχτα της Τετάρτης 3/5/2023**. Η προθεσμία είναι τελική και δεν πρόκειται να δοθεί καμία παράταση – συνολικά ή ατομικά – για κανένα λόγο.

Η παράδοση θα γίνει θα γίνει στο e-class. Για τις λεπτομέρειες της παράδοσης δείτε την παράγραφο 4.

1.3 Κώδικας Ηθικής

Κάθε ομάδα θα πρέπει να εργαστεί ανεξάρτητα. Είναι δεκτό και θεμιτό διαφορετικές ομάδες να ανταλλάξουν απόψεις σε επίπεδο γενικής ιδέας ή αλγόριθμου. Απαγορεύεται όμως κάθε συζήτηση / ανταλλαγή κώδικα ή ψευδοκώδικα. Σημειώστε ότι οι ασκήσεις θα ελεγχθούν τόσο από αυτόματο σύστημα όσο και από εμάς για ομοιότητες μεταξύ των ομάδων αλλά και για ομοιότητες με τυχόν λύσεις που μπορεί να βρεθούν στο διαδίκτυο ή λύσεις που έχουν δοθεί σε προηγούμενα έτη. Σε περίπτωση αντιγραφής οι ασκήσεις (όλων των φάσεων) όλων των εμπλεκόμενων φετινών ομάδων μηδενίζονται χωρίς καμία περαιτέρω συζήτηση.

Όλα τα μέλη στο εσωτερικό κάθε ομάδας θα πρέπει να έχουν ισότιμη συμμετοχή στην ανάπτυξη κάθε φάσης. Διατηρούμε το δικαίωμα να επιβεβαιώσουμε αν αυτό τηρείται με προσωπικές συνεντεύξεις / προφορική εξέταση.

2 Συνοπτική περιγραφή του διαχειριστή μνήμης SLOB

Ο διαχειριστής μνήμης SLOB αποθηκεύει τις σελίδες με ελεύθερο χώρο μνήμης σε τρεις λίστες. Η πρώτη λίστα χρησιμοποιείται για την εξυπηρέτηση αιτήσεων με μέγεθος μικρότερο των 256 bytes, η δεύτερη για αιτήσεις με μέγεθος μικρότερο των 1024 bytes και η τρίτη για την εξυπηρέτηση των υπολοίπων αιτήσεων. Όταν εμφανίζεται μία νέα αίτηση για δέσμευση μνήμης, ο διαχειριστής επιλέγει την κατάλληλη λίστα σελίδων βάσει του μεγέθους της αίτησης, την οποία και διατρέχει προκειμένου να εντοπίσει μία σελίδα με ελεύθερο χώρο που να μπορεί να εξυπηρετήσει την αίτηση. Για την εύρεση της κατάλληλης σελίδας, χρησιμοποιείται ο αλγόριθμος *Next-Fit*, ο οποίος αποτελεί μία παραλλαγή του αλγορίθμου *First-Fit*. Σε αντίθεση με τον αλγόριθμο *First-Fit* ο οποίος ξεκινά κάθε νέα αναζήτηση από την αρχή της λίστας, ο αλγόριθμος *Next-Fit* ξεκινά μία νέα αναζήτηση από εκείνο το σημείο της λίστας όπου έχει σταματήσει η προηγούμενη αναζήτηση. Ο αλγόριθμος επιστρέφει την πρώτη σελίδα που μπορεί να εξυπηρετήσει την αίτηση. Τέλος, σε περίπτωση που δε βρεθεί μία σελίδα, ο διαχειριστής δεσμεύει μία καινούργια σελίδας μνήμης από το λειτουργικό.

Μόλις βρεθεί μία σελίδα, ο διαχειριστής διατρέχει τα ελεύθερα blocks ώστε να εντοπίσει ένα block με αρκετό χώρο προκειμένου να εξυπηρετηθεί η αντίστοιχη αίτηση. Εντός κάθε σελίδας, τα ελεύθερα blocks οργανώνονται σε μία απλά συνδεδεμένη λίστα. Για την εύρεση του κατάλληλου block, ο διαχειριστής διαπερνά τη λίστα των ελεύθερων blocks και χρησιμοποιεί τον αλγόριθμο *First-Fit*, όπου το block που επιλέγεται είναι το πρώτο block με αρκετό ελεύθερο χώρο για την αντίστοιχη αίτηση. Σημειώστε ότι ότι η λίστα των blocks εντός της σελίδας είναι ταξινομημένη βάσει της αρχικής διεύθυνσης του block. Μόλις εντοπισθεί το κατάλληλο block, αυτό αφαιρείται από τη λίστα των ελεύθερων blocks της αντίστοιχης σελίδας και επιστρέφεται στη διεργασία που εκτέλεσε την αίτηση για δέσμευση μνήμης.

Το βασικό μειονέκτημα του διαχειριστή SLOB είναι ο μεγάλος βαθμός εξωτερικού κατακερματισμού που προκαλεί. Κύριος παράγοντας που συμβάλλει σε αυτό είναι ο αλγόριθμος *Next-Fit* και ο αλγόριθμος *First-Fit* που χρησιμοποιούνται για την διαχείριση των ελεύθερων σελίδων και των ελεύθερων blocks αντίστοιχα. Προκειμένου να μειώσετε το βαθμό εξωτερικού κατακερματισμού, θα αντικαταστήσετε και τους δύο αλγόριθμους με τον αλγόριθμο *Best-Fit*. Ο αλγόριθμος *Best-Fit* επιλέγει κάθε φορά εκείνο το τμήμα με το λιγότερο ελεύθερο χώρο, το οποίο μπορεί να εξυπηρετή-

σει την αντίστοιχη αίτηση. Ο αλγόριθμος θα χρησιμοποιηθεί τόσο για τον εντοπισμό της καταλληλότερης σελίδας όσο και για τον εντοπισμό του καταλληλότερου block εντός της σελίδας. Αν και η απόδοση και των τριών αλγορίθμων εξαρτάται σε μεγάλο βαθμό από το είδος των αιτήσεων, ο αλγόριθμος *Best-Fit* τείνει να μειώνει το βαθμό του εξωτερικού κατακερματισμού.

3 Ζητούμενα

3.1 Πρώτο Μέρος: Τροποποίηση του διαχειριστή μνήμης SLOB

3.1.1 Ενεργοποίηση του διαχειριστή μνήμης SLOB

Ο πυρήνας του Linux χρησιμοποιεί εξ' ορισμού το διαχειριστή μνήμης SLUB. Για να ενεργοποιήσετε το διαχειριστή SLOB θα πρέπει να εκτελέσετε τα ακόλουθα βήματα:

- Μεταβείτε στο φάκελο /usr/src/linux-5.4.86-dev και εκτελέστε την εντολή *sudo make menuconfig*. Στο παράθυρο που θα εμφανιστεί, μεταβείτε στην επιλογή *General Setup -> Configure standard kernel features (expert users)* και επιλέξτε την.
- Στο ίδιο παράθυρο, μεταβείτε στην επιλογή *General Setup -> Choose SLAB allocator* και επιλέξτε το διαχειριστή SLOB.
- Τέλος, μεταγλωττίστε και εγκαταστήστε το νέο πυρήνα.

Slob αργό

3.1.2 Τροποποίηση του διαχειριστή μνήμης SLOB

Η υλοποίηση του SLOB βρίσκεται στο αρχείο *mm/slob.c*. Εξοικειωθείτε με τις συναρτήσεις *slob_alloc()* και *slob_page_alloc()*. Η πρώτη συνάρτηση υλοποιεί τον αλγόριθμο *Next-Fit* για την εύρεση μίας σελίδας με επαρκές μέγεθος για να εξυπηρετήσει την τρέχουσα αίτηση, ενώ η δεύτερη συνάρτηση υλοποιεί τον αλγόριθμο *First-Fit* για τη δέσμευση του απαιτούμενου block εντός της σελίδας. Οι αλλαγές που θα πραγματοποιήσετε αφορούν αυτές τις δύο συναρτήσεις, στις οποίες θα πρέπει να αντικαταστήσετε τους δύο αλγόριθμους με τον αλγόριθμο *Best-Fit*.

Ένα από τα ζητούμενα της εργασίας είναι η σύγκριση μεταξύ των αλγορίθμων *First-Fit* και *Best-Fit*. Γι' αυτό θα πρέπει να αναπτύξετε τον κώδικά σας με τέτοιο τρόπο ώστε να επιτρέπεται να ενεργοποιείτε και να απενεργοποιείτε την εκάστοτε υλοποίηση με εύκολο τρόπο. Μία προσέγγιση είναι η χρήση των εντολών του προεπεξεργαστή για μετάφραση κώδικα υπό συνθήκη (π.χ. εντολή #ifdef).

3.1.3 Εκτύπωση στοιχείων εκτέλεσης αλγορίθμου Best-Fit

Το δεύτερο βήμα στη διαδικασία τροποποίησης του διαχειριστή μνήμης είναι να προσθέσετε κώδικα για την εμφάνιση στοιχείων εκτέλεσης του αλγορίθμου *Best-Fit* κατά τη δέσμευση ενός block. Δηλαδή, ο κώδικας για την εκτύπωση των στοιχείων εκτέλεσης θα πρέπει να προστεθεί μόνο στη συνάρτηση *slob_page_alloc()*. Τα στοιχεία που θα πρέπει να εκτυπώνονται κατά την εκτέλεση του αλγορίθμου είναι το **μέγεθος της αίτησης**, το **μεγέθος ελεύθερου χώρου** του εκάστοτε block και το **μεγέθος ελεύθερου χώρου** του block που επιλέγεται προς δέσμευση από τον αλγόριθμο. Όλα τα μεγέθη θα πρέπει να εκτυπώνονται σε μονάδες *SLOB_UNITS*.

Για την εκτύπωση των μηνυμάτων θα χρησιμοποιήσετε την συνάρτηση *printk()* ενώ για να μπορείτε να δείτε τα μηνύματα που εκτυπώνονται θα χρησιμοποιήσετε την εντολή **dmesg** σε κάποιο terminal. Προσθέστε στην αρχή κάθε μηνύματος το κείμενο *slob_alloc:*, ώστε να είναι εύκολη η αναζήτηση των αντίστοιχων μηνυμάτων. Παρακάτω παρουσιάζονται ενδεικτικά στιγμιότυπα εκτέλεσης:

Not a all

```
slob_alloc: Request: 106
slob_alloc: Candidate blocks size: 28 4 4 30 6 2 30 38 4 6 26 8 10 114
slob_alloc: Best Fit: 114

slob_alloc: Request: 74
slob_alloc: Candidate blocks size: 20 54 0 0 22 0 0 0 0 0 0 0 0 26 0 0 0 0
slob_alloc: Best Fit: None

slob_alloc: Request: 92
slob_alloc: Candidate blocks size: 236 92 24 24 184 128 92 120 24 256 28
slob_alloc: Best Fit: 92
```

Όπως μπορείτε να δείτε στο δεύτερο στιγμιότυπο εκτέλεσης, στην περίπτωση που δεν βρεθεί κάποιο block που μπορεί να εξυπηρετήσει την αίτηση θα πρέπει να εκτυπώνετε slob_alloc: Best Fit: None. Επίσης στο τρίτο στιγμιότυπο εκτέλεσης, όπου υπάρχει block με το ακριβές μέγεθος, η αναζήτηση θα μπορούσε να σταματήσει μόλις αυτό εντοπιζόταν.

Καθώς η συνεχής εκτύπωση μηνυμάτων θα έχει ως αποτέλεσμα τη διόγκωση των log files, αρκεί να εκτυπώνετε τα μηνύματα κατά περιοδικά διαστήματα. Για παράδειγμα, η εκτύπωση θα μπορούσε να πραγματοποιείται κάθε 6000 κλήσεις της συνάρτησης *slob_alloc()*. Για να το επιτύχετε αυτό,

ορίστε μία μεταβλητή με τύπο *static unsigned int* στο αρχείο slob.c, η οποία θα αυξάνεται κατά 1 σε κάθε κλήση της slob_alloc(). Όταν η τιμή της γίνει ίση με την προκαθορισμένη συχνότητα, θα εκτυπώνετε τα απαραίτητα μηνύματα και θα αρχικοποιείτε εκ νέου τη μεταβλητή στην τιμή 0.

3.1.4 Υλοποίηση κλήσεων συστήματος για τη συγκέντρωση στατιστικών στοιχείων

Το επόμενο βήμα στην υλοποίησή σας είναι η υλοποίηση δύο κλήσεων συστήματος που θα επιτρέπουν τη συγκέντρωση στατιστικών στοιχείων που αφορούν στην εκτέλεση του διαχειριστή μνήμης. Επινοήστε μία μέθοδο ώστε να είναι δυνατή η συγκέντρωση του συνολικού μεγέθους της μνήμης που έχει δεσμευθεί από το διαχειριστή καθώς και του μεγέθους της μνήμης που δε χρησιμοποιήθηκε κατά την εξυπηρέτηση μίας αίτησης για δέσμευση μνήμης. Για το σκοπό αυτό θα χρησιμοποιήσετε δύο *global* και *static* μεταβλητές που θα αποθηκεύουν τα δύο μεγέθη, τις οποίες και θα ενημερώνετε σε κατάλληλα σημεία. Για παράδειγμα, για το συνολικό μέγεθος της μνήμης που έχει δεσμευθεί από το διαχειριστή, μπορείτε να ενημερώνετε την αντίστοιχη μεταβλητή στις συναρτήσεις *slob_new_pages()* και *slob_free_pages()* οι οποίες καλούνται κατά τη δέσμευση και αποδέσμευση αντίστοιχα μίας σελίδας μνήμης.

Όσον αφορά στον υπολογισμό του μεγέθους της μνήμης που δε χρησιμοποιήθηκε κατά την εξυπηρέτηση μίας αίτησης, θα πρέπει να διατρέχετε και τις τρεις λίστες όπου αποθηκεύονται οι ελεύθερες σελίδες και να αθροίζετε το σύνολο της ελεύθερης μνήμης σε κάθε σελίδα. Το σύνολο του ελεύθερου χώρου σε μία σελίδα αποθηκεύεται στο πεδίο *units* της δομής *struct page* στο αρχείο linux/include/mm_types.h, η οποία περιγράφει μία σελίδα μνήμης. Τέλος, η συνάρτηση *slob_alloc()* αποτελεί ένα πιθανό σημείο όπου μπορεί να πραγματοποιηθεί η ενημέρωση της αντίστοιχης μεταβλητής.

Έπειτα υλοποιήστε δύο κλήσεις συστήματος που θα επιστρέφουν αυτά τα μεγέθη σε bytes, ως ακεραίους τύπου *long*, με τα παρακάτω πρότυπα:

- long slob_get_total_alloc_mem(void)
- long slob_get_total_free_mem(void)

Χρησιμοποιώντας αυτές τις συναρτήσεις μπορείτε να έχετε μία προσέγγιση του μεγέθους του εξωτερικού κατακερματισμού του εκάστοτε αλγορίθμου δέσμευσης μνήμης.

3.1.5 Μεταγλώττιση του νέου πυρήνα

Ως τελευταίο βήμα, θα πρέπει να μεταγλωττίσετε και να εγκαταστήσετε το νέο πυρήνα. Εκτελέστε την αντίστοιχη διαδικασία και επανεκκινήστε το νέο πυρήνα, όπου πλέον ο διαχειριστής SLOB χρησιμοποιεί τον αλγόριθμο *Best-Fit*. Τι παρατηρείτε σε σύγκριση με τον αλγόριθμο *First-Fit*?

3.2 Πειραματική αξιολόγηση

Για να αξιολογήσετε την επίδοση του αλγορίθμου *Best-Fit*, θα πρέπει να προκαλέσετε την ενεργοποίηση του αλγόριθμου και ακολούθως να παρακολουθήσετε τα στατιστικά της συμπεριφοράς του.

Για την ενεργοποίηση του αλγόριθμου υπάρχουν δύο επιλογές:

1. Να χρησιμοποιήσετε ενεργά το σύστημά σας για κάποιο χρονικό διάστημα, εκτελώντας διάφορες δραστηριότητες, αξιοποιώντας σε κάποια πειράματα τον αλγόριθμο *Best-Fit* και σε κάποια τον *First-Fit*. Η επιλογή αυτή είναι η ευκολότερη, αλλά έχει το προφανές μειονέκτημα ότι είναι αδύνατο να εκτελέσετε πιστά την ίδια ακολουθία δραστηριοτήτων κατά τη χρήση καθενός από τους δύο αλγόριθμους.

Αυτό είναι πιο Καλό

2. Να υλοποιήσετε δύο system calls, ένα που θα ενεργοποιεί την εκτέλεση kmalloc (για μέγεθος μνήμης που θα καθορίζετε ως παράμετρο στο system call) και θα επιστρέφει τη διεύθυνση που δεσμεύθηκε[2], και ένα που θα παίρνει ως παράμετρο κατά την κλήση του μία διεύθυνση και ενεργοποιεί την kfree για την περιοχή μνήμης που ξεκινά από αυτή τη διεύθυνση (και είχε νωρίτερα δεσμευθεί με την προηγούμενη κλήση συστήματος). Ακολούθως, θα πρέπει να υλοποιήσετε εφαρμογή επιπέδου χρήστη η οποία θα χρησιμοποιεί τα system calls για να επιτύχει μία σειρά από δεσμεύσεις / αποδεσμεύσεις με διαφορετικά μεγέθη και με τη σειρά που θα επιλέξετε. Το ίδιο πείραμα θα πρέπει να εκτελεστεί διαδοχικά με ενεργό τον *Best-Fit* και κατόπιν τον *First-Fit*.

Η 2η επιλογή προσφέρει σαφώς καλύτερο έλεγχο της πειραματικής διαδικασίας.

Για τη λήψη των στατιστικών, θα πρέπει να αναπτύξετε μία απλή εφαρμογή ώστε να συγκρίνετε το μέγεθος του εξωτερικού κατακερματισμού του διαχειριστή SLOB πριν και μετά την τροποποίηση. Η **μόνη** λειτουργικότητα που θα παρέχεται από την εφαρμογή σας είναι η χρήση των δύο

[2]Σημειώστε ότι αυτή η διεύθυνση σας είναι άχρηστη στο επίπεδο χρήστη. Θα τη χρησιμοποιήσετε μόνο ως παράμετρο για τη μετέπειτα απελευθέρωση αυτής της μνήμης.

κλήσεων συστήματος συλλογής στατιστικών που αναπτύξατε. Εκτελέστε την εφαρμογή σας και με τους δύο αλγορίθμους για έναν ικανό αριθμό από επαναλήψεις προκειμένου να λάβετε ένα μέσο όρο για το συνολικό μέγεθος της μνήμης που έχει δεσμευθεί καθώς και για το μέγεθος της ελεύθερης μνήμης, η οποία δε χρησιμοποιείται κατά την εξυπηρέτηση μίας αίτησης. Στη συνέχεια καταγράψτε τις παρατηρήσεις σας σε ένα αρχείο κειμένου. Στο ίδιο αρχείο καταγράψτε και τις παρατηρήσεις της ενότητας 3.1.5.

4 Πακετάρισμα των Αλλαγών – Δημιουργία του προς Υποβολή Συνημμένου Αρχείου

4.1 Απομόνωση των Αλλαγών στον Κώδικα του Λειτουργικού

Προφανώς δεν είναι ιδιαίτερα πρακτικό να παραδώσετε τις αλλαγές στέλνοντας όλο το νέο κώδικα του λειτουργικού, δεδομένου και ότι τα αρχεία τα οποία στην πραγματικότητα θα έχετε πειράξει είναι ελάχιστα. Η εφαρμογή *diff* αναλαμβάνει να αναγνωρίσει τις αλλαγές που κάνατε στον πηγαίο κώδικα του λειτουργικού, συγκρίνοντάς τον με το αντίγραφο του αρχικού κώδικα που έχουμε κρατήσει στον κατάλογο /usr/src/linux-5.4.86-orig.

Κατ’ αρχήν πηγαίνετε στον κατάλογο /usr/src/linux-5.4.86-dev (cd /usr/src/linux-5.4.86-dev) και δώστε την εντολή *make distclean*. Με τον τρόπο αυτό σβήνονται όλα τα αρχεία (.o, εκτελέσιμα κλπ) που δημιουργήθηκαν κατά τη μεταγλώττιση. Κάνετε το ίδιο – για καλό και για κακό – και στον κατάλογο /usr/src/linux-5.4.86-orig. **ΜΗΝ ΞΕΧΑΣΕΤΕ ΑΥΤΑ ΤΑ 2 ΒΗΜΑΤΑ!!!**

Πλέον είστε έτοιμοι να “συγκρίνετε” τους 2 καταλόγους. Πηγαίνετε στο /usr/src και από εκεί δώστε την εντολή

sudo bash -c ’diff -ruN linux-5.4.86-orig linux-5.4.86-dev > patch_3’

Το πρόγραμμα diff εντοπίζει αναδρομικά τις αλλαγές μεταξύ των καταλόγων linux-5.4.86-orig και linux-5.4.86-dev. Το τελικό αποτέλεσμα αποθηκεύεται στο αρχείο patch_3 το οποίο δημιουργείται μέσα στον κατάλογο από τον οποίο καλέσατε την *diff* (δηλαδή τον κατάλογο /usr/src). Αν θέλετε, διαβάστε το αρχείο patch_3 (είναι ένα αρχείο κειμένου) και προσπαθήστε να καταλάβετε τη δομή του.

Αν θέλετε να επιβεβαιώσετε ότι το patch φτιάχτηκε σωστά, μπορείτε να κάνετε τα ακόλουθα: Η εφαρμογή *patch* μπορεί να πάρει ένα patch και να εφαρμόσει τις αλλαγές που περιγράφει το patch

σε έναν κατάλογο πηγαίου κώδικα. Κατόπιν, μεταβείτε στον κατάλογο /usr/src/linux-5.4.86-orig και δώστε την εντολή

patch -p1 –dry-run < ../patch_3

Με την εντολή αυτή θα γίνει προσομοίωση εφαρμογής στον κατάλογο που βρισκόμαστε του patch που περιέχεται στο αρχείο patch_3 (το οποίο patch_3 είναι τοποθετημένο στον στον αμέσως προηγούμενο κατάλογο, γι’ αυτό και το ../). Η παράμετρος *–dry-run* λέει στην εφαρμογή patch να προσποιηθεί ότι εφαρμόζει το patch χωρίς να κάνει στην πραγματικότητα οποιεσδήποτε αλλαγές στα αρχεία. Το patch θα πρέπει να εφαρμοστεί “καθαρά”, δε θα πρέπει δηλαδή να σας εμφανιστούν μηνύματα λάθους. Δώστε προσοχή σε αυτό το βήμα, γιατί προϋπόθεση για τη βαθμολόγηση κάθε άσκησης είναι το patch που θα μας στείλετε να μπορεί να εφαρμοστεί “καθαρά”.

4.2 Πακετάρισμα των Προς Υποβολή Στοιχείων σε Αρχείο

Το τελευταίο βήμα είναι να πακετάρετε όλα τα αρχεία που πρέπει να μας στείλετε σε ένα αρχείο. Πηγαίνετε στον home directory του λογαριασμού σας. Φτιάξτε εκεί με την εντολή *mkdir* έναν κατάλογο με όνομα project_3_omada_<AEM_omadas>. Για παράδειγμα, η ομάδα με μέλη με ΑΕΜ 456 123 789 θα πρέπει να δώσει την εντολή *mkdir project_3_omada_456_123_789*. Αντιγράψτε (με την εντολή *cp*) το patch που φτιάξατε σε αυτό τον κατάλογο με την εντολή. Π.χ. η ομάδα 15 θα δώσει την εντολή: *cp /usr/src/patch_3 /project_3_omada_456_123_789*.

Κάντε το ίδιο και με τα αρχεία που απαρτίζουν τη βιβλιοθήκη επιπέδου χρήστη για τη χρήση των κλήσεων συστήματος, το αρχείο με τον κώδικα της εφαρμογής που αναπτύξατε καθώς και το αρχείο κειμένου με τις παρατηρήσεις σας.

Επίσης, δημιουργήστε μέσα στον κατάλογο ένα αρχείο με όνομα README.txt στον οποίο θα γράψετε τα ονόματα, ΑΕΜ και e-mail των μελών της ομάδας, καθώς και αναλυτικές οδηγίες για τη μεταγλώττιση της πειραματικής εφαρμογής. Μπορείτε να συμπεριλάβετε στο αρχείο και οτιδήποτε άλλο θέλετε να έχουμε υπόψη μας κατά τη διόρθωση.

Ακολούθως, πηγαίνετε στον πηγαίο κατάλογο του λογαριασμού σας και από εκεί δώστε την εντολή

tar -cvf project_3_omada_<AEM_omadas>.tar project_3_omada_<AEM_omadas>

Για παράδειγμα, η ομάδα με μέλη με ΑΕΜ 456 123 789 θα δώσει την εντολή

tar -cvf project_3_omada_123_456_789.tar project_3_omada_123_456_789

Με την εντολή tar θα πακεταριστούν τα περιεχόμενα του καταλόγου σε ένα αρχείο με κατάληξη .tar.

Τέλος, δώστε την εντολή *bzip2 project_3_omada_<arithmos_omadas>.tar*. Θα δημιουργηθεί ένα αρχείο με κατάληξη .bz2, το οποίο περιέχει συμπιεσμένο το αρχείο με κατάληξη .tar. Το αρχείο με κατάληξη .bz2 είναι αυτό που θα πρέπει να ανεβάσετε στο e-class. Για παράδειγμα, η ομάδα η ομάδα με μέλη με ΑΕΜ 456 123 789 θα δώσει την εντολή *bzip2 project_3_omada_123_456_789.tar*.

Αφού φτιαχτεί το αρχείο, ελέγξτε το μέγεθός του. Αν το μέγεθος είναι αδικαιολόγητα μεγάλο (θυμηθείτε ότι το αρχείο στην ουσία περιέχει μερικά πολύ μικρά αρχεία κειμένου) έχετε κάνει κάτι λάθος (το πιθανότερο είναι ότι έχετε ξεχάσει το βήμα make distclean ή έχετε συμπεριλάβει και εκτελέσιμα αρχεία).