Λειτουργικά Συστήματα, Κ22
Χειμερινό Εξάμηνο ‘17-’18
Τμήμα Ζυγών Α/Μ (Γ. Σμαραγδάκης)
Άσκηση 1
Προθεσμία 29/10/17

Σκοπός της Εργασίας

Καλείστε να υλοποιήσετε έναν απλό memory allocator, δηλαδή μια βιβλιοθήκη που θα μπορεί να χρησιμοποιηθεί σαν εναλλακτική των κλήσεων malloc/free. Ο allocator χρησιμοποιεί τη στρατηγική που λέγεται segregated fit (δηλαδή χωριστή αποθήκευση ανά κλάση μεγέθους αντικειμένου).

Συγκεκριμένα θα υλοποιήσετε δύο ρουτίνες mymalloc/myfree οι οποίες θα χειρίζονται αιτήματα δέσμευσης μνήμης μέχρι 4096 bytes. Οι αλγόριθμοι που θα υλοποιήσετε περιγράφονται παρακάτω.

Κλήσεις στη λειτουργία "void *mymalloc(int cbytes)" (ή, πιο σωστά, “void *mymalloc(size_t cbytes)”) θα έχουν την εξής λογική:

- Αν cbytes > 4096, καλέστε την κανονική malloc και επιστρέψτε ό,τι επιστρέψει.
- Αν cbytes <= 4096, στρογγυλέψτε τον αριθμό στην κοντινότερη μεγαλύτερη δύναμη του 2 που είναι τουλάχιστον 32 bytes. (Δηλαδή υπάρχουν 8 *κλάσεις μεγέθους*: 32, 64, 128, 256, 512, 1024, 2048, ή 4096 bytes.)
  Παράδειγμα:
  mymalloc(32) => επιστρέψτε χώρο 32 bytes
  mymalloc(5) => επιστρέψτε χώρο 32 bytes
  mymalloc(73) => επιστρέψτε χώρο 128 bytes.

Mymalloc

Η καθεμία από τις 8 κλάσεις μεγέθους αντικειμένου που χειρίζεστε θα έχει μια (διπλά συνδεδεμένη) λίστα που δείχνει τις *σελίδες* που χρησιμοποιεί για αποθήκευση αντικειμένων. (Μπορείτε δηλαδή να έχετε ένα array 8 θέσεων που να δείχνουν στις αντίστοιχες κεφαλές των 8 λιστών.) Οι σελίδες αποτελούνται από 4096 bytes και είναι ευθυγραμμισμένες στα 4096 bytes, δηλαδή η διεύθυνση της κάθε σελίδας είναι πολλαπλάσιο του 4096. Η κάθε σελίδα δεσμεύεται να αποθηκεύει μόνο αντικείμενα ενός μεγέθους. Π.χ. μια σελίδα που βρίσκεται στη λίστα της κλάσης μεγέθους “256 bytes” μπορεί να αποθηκεύσει μέχρι 16 αντικείμενα των (μετά τη στρογγύλευση) 256 bytes.

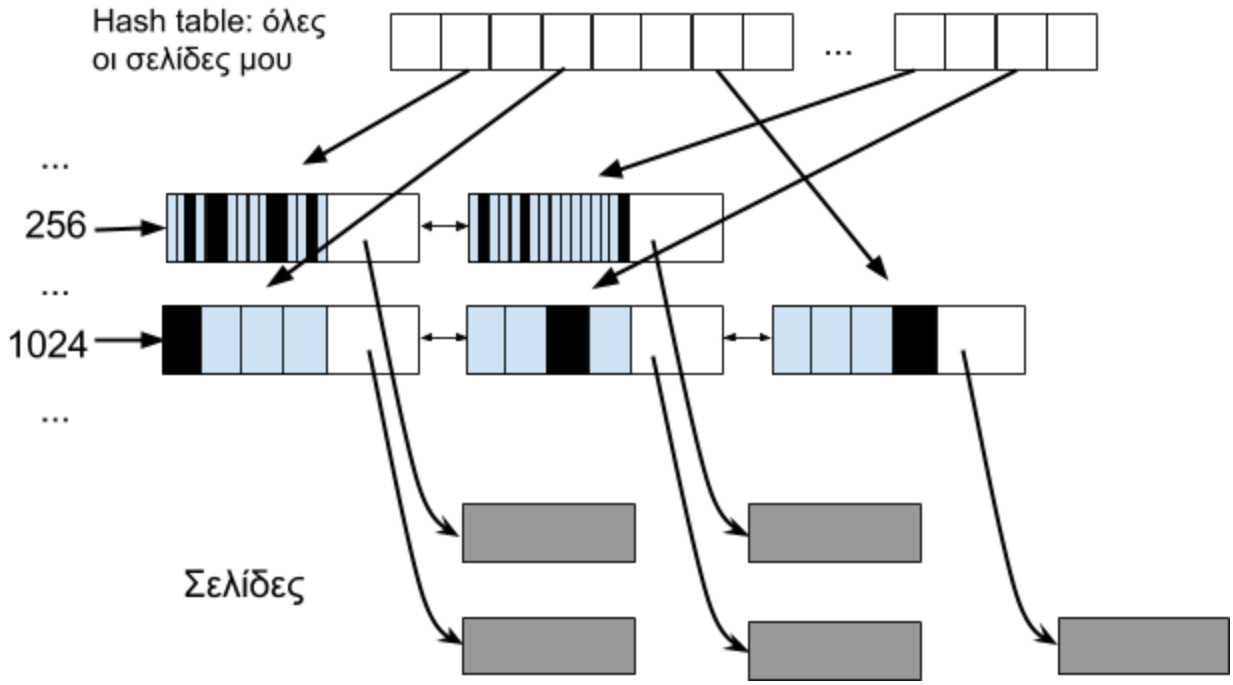
Αντίστοιχη σελίδα στη λίστα της κλάσης μεγέθους 2048 bytes μπορεί να αποθηκεύσει μέχρι 2 αντικείμενα (των 2048 bytes).

Σελίδες παίρνετε από το λειτουργικό σύστημα κάνοντας ένα μεγάλο malloc (τουλάχιστον 1MB) και “κόβοντας” σε σελίδες το χώρο από το πρώτο έως το τελευταίο πολλαπλάσιο του 4096 στη μνήμη που θα επιστραφεί. Ο χώρος που παραμένει πριν την πρώτη και μετά την τελευταία σελίδα δεν θα χρησιμοποιηθεί. Με αυτό τον τρόπο θα πάρετε κάθε φορά από την malloc του συστήματος μερικές εκατοντάδες σελίδες, την *αποθήκη σελίδων* σας. Όταν μια κλάση μεγέθους χρειάζεται να βρει χώρο για νέο αντικείμενο και δεν έχει τέτοιο χώρο στις υπάρχουσες σελίδες της, παίρνει **μία** νέα σελίδα από την αποθήκη σελίδων. Δεν υπάρχει τρόπος επιστροφής σελίδων στην αποθήκη σελίδων: οι σελίδες που μπαίνουν σε λίστα για συγκεκριμένη κλάση μεγέθους μένουν εκεί.

Σε κάθε mymalloc καλείστε να βρείτε μνήμη σε μια σελίδα. Για να γίνει αυτό, η λίστα της κάθε κλάσης μεγέθους κρατάει ανά σελίδα και ένα bit vector που δείχνει ποιες θέσεις είναι ελεύθερες. Αν καμία θέση δεν είναι ελεύθερη στην πρώτη σελίδα της λίστας, εξετάζετε την επόμενη, κ.ο.κ. (Όπως αναφέρθηκε ήδη, αν η λίστα εξαντληθεί, παίρνετε σελίδα από την αποθήκη σελίδων. Αν και αυτή είναι άδεια, φτιάχνετε νέα αποθήκη με μεγάλο malloc από το σύστημα.) Όταν βρεθεί σελίδα με ελεύθερη θέση, τη δεσμεύετε και μετακινείτε τη σελίδα στην αρχή της λίστας, ώστε μελλοντικές αναζητήσεις να είναι πιο γρήγορες.

Τέλος, όλοι οι κόμβοι (που αντιστοιχούν σε σελίδες) σε οποιαδήποτε λίστα κλάσης μεγέθους μπαίνουν σε ένα hash table (πίνακα διασποράς/κατακερματισμού) ή άλλη γρήγορη δομή αναζήτησης (π.χ. δυαδικό δέντρο) ώστε να μπορεί κανείς γρήγορα να βρει αν μια σελίδα (με βάση τη διεύθυνσή της) χρησιμοποιείται από τη mymalloc. (Αυτό θα σας χρειαστεί στο myfree, αλλά δίνει και ένα από τα βασικά πλεονεκτήματα αυτού του είδους allocator: από οποιοδήποτε δείκτη, ακόμα και στη μέση ενός αντικειμένου, μπορούμε να βρούμε σε σταθερό χρόνο την αρχή και το μέγεθος του αντικειμένου, στρογγυλεύοντας τη διεύθυνση για να βρούμε την αρχή σελίδας.) Η στρατηγική για το hash table (μέγεθος, συναρτήσεις hash, κτλ.) ή άλλη γρήγορη δομή είναι δική σας επιλογή.

Το παρακάτω σχήμα απεικονίζει τις βασικές δομές.



Σχήμα 1: Οι βασικές δομές: λίστες, για 2 ενδεικτικές κλάσεις μεγέθους, με bit vectors που δείχνουν πού υπάρχουν αντικείμενα ανά σελίδα, καθώς και hash table με όλες τις σελίδες που χρησιμοποιούνται.

Myfree

Έχουμε δύο ειδών αντικείμενα που ελευθερώνονται: αυτά που ανήκουν στη malloc του συστήματος και αυτά που ανήκουν στο δικό μας allocator. Για να βρούμε το είδος του αντικειμένου, στρογγυλεύουμε τη διεύθυνσή του (προς τα κάτω) σε πολλαπλάσιο του 4096 και κοιτάμε αν αυτή αντιστοιχεί σε σελίδα του allocator μας. Αν το αντικείμενο δεν είναι σε δική μας σελίδα, απλά καλούμε τη free του συστήματος. Αλλιώς, ανανεώνουμε το αντίστοιχο bit vector. Σε κάποια δομή (είτε σε κάθε κόμβο λίστας είτε σε κάθε entry του hash table) θα χρειαστεί να κρατήσετε και κάποιο συσχετισμό με την κλάση μεγέθους, έτσι ώστε από ένα κόμβο λίστας να μπορεί κανείς άμεσα να ξέρει το μέγεθος του αντικειμένου και τη διάρθρωση του bit vector. Το πώς ακριβώς θα γίνει αυτό είναι δική σας σχεδιαστική επιλογή.

Δοκιμές

Θα χρησιμοποιήσετε τον allocator σας μαζί με νέο κώδικα δοκιμασίας που θα γράψετε εσείς, καθώς και μαζί με τουλάχιστον μία παλιότερη άσκησή σας από προηγούμενα μαθήματα, που να χρησιμοποιεί δυναμικές δομές. Δηλαδή θα πάρετε τον κώδικά σας από προηγούμενο μάθημα και όπου χρησιμοποιούσε malloc/free θα τον αλλάξετε ώστε να χρησιμοποιεί mymalloc/myfree. Αυτό θα είναι μέρος της επίδειξης της άσκησης

στους βοηθούς. Θα πρέπει να γνωρίζετε και τον παλιότερο κώδικα για εξήγηση αν χρειαστεί.

Διαδικαστικά

- Το πρόγραμμά σας θα πρέπει να γραφτεί σε C/C++ και να τρέχει στις μηχανές Linux workstations (linuxXY.di.uoa.gr ή linuxvmXY.di.uoa.gr) του τμήματος.
- Οι δομές (λίστα + γρήγορη δομή αναζήτησης) θα υλοποιηθούν από την αρχή για τους σκοπούς της άσκησης. Φυσικά μπορείτε να χρησιμοποιήσετε **δικό σας** παλιότερο κώδικα αν α) δεν γίνεται κάτι καλύτερο στην άσκηση αυτή, δηλαδή απλά θα ξαναγράφατε τον ίδιο κώδικα, β) ξέρετε τον κώδικα πλήρως και μπορείτε να τον εξηγήσετε και ξαναγράψετε όποτε ζητηθεί κατά την εξέταση.
- Η χρήση Makefile είναι επιτακτική!
- Παρακολουθείτε την ιστοσελίδα του μαθήματος http://yanniss.github.io/k22/ για επιπρόσθετες ανακοινώσεις αλλά και την ηλεκτρονική-λίστα (η-λίστα) του μαθήματος στο URL https://piazza.com/uoa.gr/fall2017/k22/home .
- Αν ο σχεδιασμός των αρχείων σας είναι ερασιτεχνικός (π.χ. όλες οι δομές σε ένα αρχείο, λάθος διαχωρισμός header και .c files) θα υπάρξει βαθμολογική ποινή.

Τι Πρέπει να Παραδοθεί

1. Μια σύντομη και περιεκτική εξήγηση για τις επιλογές που έχετε κάνει στο σχεδιασμό του προγράμματος σας (1 σελίδα σε ASCII κείμενο είναι αρκετή).
2. Ένα Makefile (που να μπορεί να χρησιμοποιηθεί για να γίνει αυτόματα το compile του προγράμματος σας).
3. ΄Ενα tar-file με όλη σας την δουλειά σε έναν κατάλογο που πιθανώς να φέρει το όνομα σας και θα περιέχει όλη σας την δουλειά δηλ. source files, header files, output files (αν υπάρχουν) και οτιδήποτε άλλο χρειάζεται.

΄Αλλες Σημαντικές Παρατηρήσεις

1. Οι εργασίες είναι **ατομικές**.
2. Αν και αναμένεται να συζητήσετε με φίλους/συμφοιτητές το πως θα επιχειρήσετε να δώσετε λύση στο πρόβλημα, αντιγραφή κώδικα (οποιαδήποτε μορφής) **δεν επιτρέπεται**. Οποιοσδήποτε βρεθεί αναμεμειγμένος σε αντιγραφή κώδικα **μηδενίζεται** στο μάθημα. Αυτό ισχύει για όσους εμπλέκονται ανεξάρτητα από το ποιος έδωσε/πήρε κλπ.
3. Αν πιθανώς κάποια (δευτερεύοντα!) κομμάτια του κωδικά σας προέλθουν από κάποια δημόσια πηγή, θα πρέπει να δώσετε αναφορά στη εν λόγω πηγή είτε αυτή είναι βιβλίο, σημειώσεις, Internet URL κλπ. και να εξηγήσετε πως ακριβώς χρησιμοποιήσατε την εν λόγω αναφορά.