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

Άσκηση 2

Προθεσμία 27/11/17

Σκοπός της Εργασίας

Στη δεύτερη εργασία του μαθήματος, καλείστε να παραμετροποιήσετε το λειτουργικό σύστημα xv6 προκειμένου να υλοποιήσετε έναν δικό σας scheduler και μια κλήση συστήματος που θα τυπώνει κάποια στοιχεία για τις διεργασίες που υπάρχουν μέσα στον scheduler σας.

Πιο συγκεκριμένα, η άσκηση χωρίζεται σε 3 μέρη:

- Υλοποίηση μικρών προγραμμάτων που δημιουργούν φόρτο εργασίας (workload) για δοκιμές στον scheduler που θα υλοποιήσετε,
- **Κύριο μέρος της άσκησης**: Έναν min-heap scheduler,
- Το system-call printRunningProc, το οποίο θα τυπώνει όλα τα processes τα οποία υπάρχουν στον scheduler σας τη στιγμή της εκτέλεσης.

Υλοποίηση δοκιμαστικών Workload

Σε αυτό το πρώτο κομμάτι της άσκησης, σκοπός σας είναι να υλοποιήσετε 5 εκτελέσιμα προγράμματα το καθένα με τις δικές του ιδιαιτερότητες.

- Το πρώτο θα είναι ένα απλό πρόγραμμα το οποίο θα εκτελεί πολλαπλούς υπολογισμούς. Μια επαναληπτική αριθμητική πράξη, η οποία θα εκτελείται για αρκετή ώρα (π.χ. λεπτά) σε πραγματικό χρόνο, είναι αρκετή.
- Το δεύτερο θα είναι ένα πρόγραμμα το οποίο θα εκτελεί επαναληπτικά και σύντομα I/O. Πιο συγκεκριμένα, θα διαβάζει έναν - έναν τους χαρακτήρες από ένα αρχείο της επιλογής σας (πιθανόν standard input). Ο σκοπός είναι να τρέχετε το πρόγραμμα αυτό πάλι για αρκετή ώρα (δηλ. λεπτά), χωρίς ανθρώπινη παρέμβαση, με τις κατάλληλες παραμέτρους. (Φυσικά μπορεί και χωρίς καμία παράμετρο αν το φτιάξετε έτσι ώστε να αναφέρεται σε συγκεκριμένο αρχείο.)
- Το τρίτο πρόγραμμα ακολουθεί το ίδιο σκεπτικό με το δεύτερο με την θεμελιακή διαφορά ότι τα I/Os θα είναι μεγάλα (π.χ. χιλιάδες χαρακτήρες ανά I/O).
- Στη συνέχεια, το 4ο πρόγραμμά σας θα υλοποιεί την ίδια λειτουργία με το 2ο, με τη διαφορά ότι ενδιάμεσα στα I/O, θα γίνονται και πολλαπλοί υπολογισμοί. Θα υπάρχει δηλαδή χρονική απόσταση ανάμεσα σε κάθε I/O, λόγω του ότι η CPU είναι απασχολημένη με εργασία εντός του process.
- Τέλος, το 5ο και τελευταίο πρόγραμμα σας, θα υλοποιεί την ίδια λειτουργία με το 4ο, με τη διαφορά ότι τα I/Os θα είναι μεγάλα και όχι ανά έναν χαρακτήρα.

Κύριο Μέρος: Υλοποίηση του Scheduler

Στο 2ο κομμάτι της εργασίας σας, σκοπός είναι να υλοποιήσετε έναν scheduler που θα βασίζεται σε μια δομή min-heap. (Η δομή γενικά λέγεται απλά “heap” και min-heap είναι μόνο η συγκεκριμένη παραλλαγή που περιγράφουμε. Στην άσκηση αυτή γράφουμε πάντα “min-heap” για να μην υπάρξει σύγχυση με το γνωστό συνώνυμο χώρο μνήμης “heap” κάθε διεργασίας.)

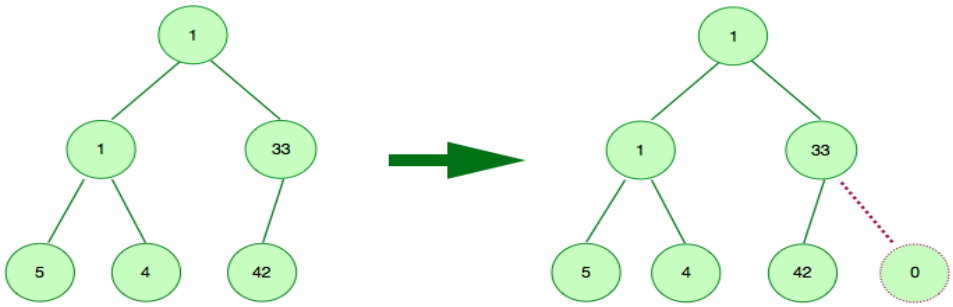
Αναλυτικότερα, το min-heap είναι ένα δυαδικό δέντρο (binary tree) που ΔΕΝ είναι πλήρως διατεταγμένο (δηλαδή ΔΕΝ είναι binary search tree) αλλά έχει μια απλή ιδιότητα: ο κάθε κόμβος έχει τιμή μικρότερη ή ίση από τις τιμές των παιδιών του.
(***Σημείωση***: *Περισσότερες πληροφορίες για τη δομή min-heap μπορείτε να βρείτε εδώ:* **https://en.wikipedia.org/wiki/Heap_(data_structure)**)

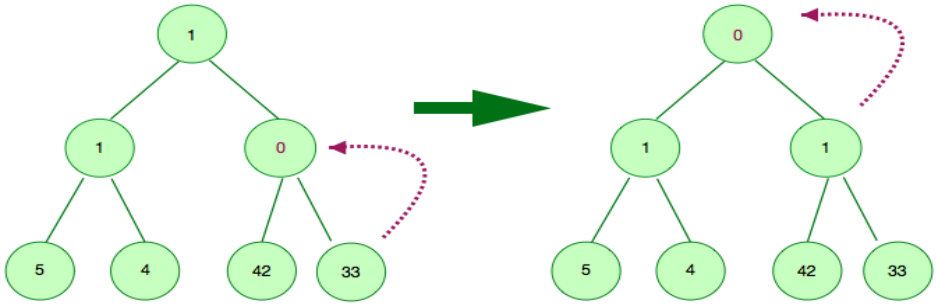
Η δομή αυτή θα χρησιμοποιηθεί ως priority queue για το χρονοπρογραμματισμό των διεργασιών στο σύστημα. Δηλαδή κάθε κόμβος του min-heap θα αντιπροσωπεύει μια διεργασία σε κατάσταση READY και την τρέχουσα προτεραιότητά της. Όταν το λειτουργικό σύστημα θα θελήσει να διαλέξει διεργασία για να εκτελεστεί (δηλαδή να πάει από κατάσταση READY σε RUNNING) θα διαλέξει απλά τη διεργασία με τη μικρότερη τρέχουσα προτεραιότητα, δηλαδή τη ρίζα του min-heap.

Λεπτομέρειες δομής min-heap: Η δομή min-heap που θα υλοποιήσετε θα έχει το πολύ 64 κόμβους - όσο το μέγιστο πλήθος διεργασιών στο xv6. Άρα μπορείτε να την αποθηκεύσετε σε στατικό array 64 θέσεων. Αλλά η δομή κατανοείται πιο εύκολα σαν δέντρο.

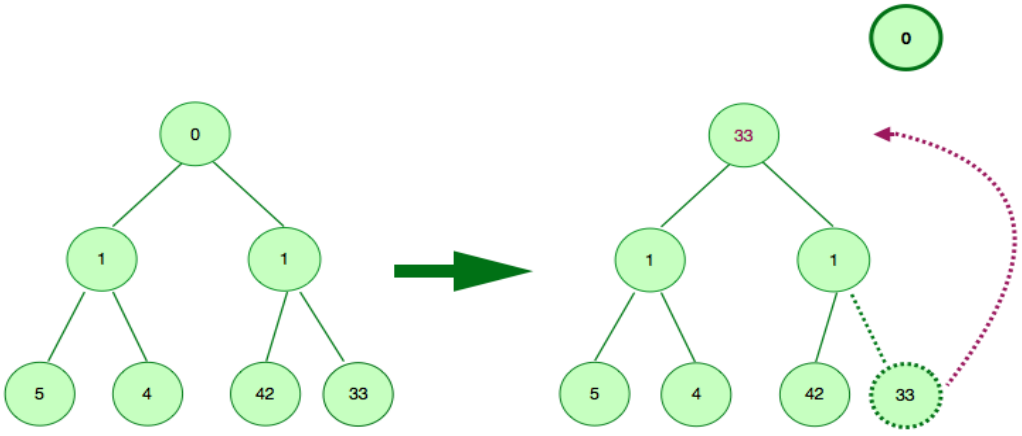
Θα διατηρήσετε την ιδιότητα ότι το min-heap είναι δυαδικό δέντρο πάντα γεμάτο σε κάθε επίπεδο εκτός από το τελευταίο του, και το τελευταίο επίπεδο γεμίζει από αριστερά προς τα δεξιά. Κάθε στοιχείο εισάγεται αρχικά στο τελευταίο επίπεδο του min-heap. Αν η τιμή του είναι μεγαλύτερη ή ίση από την τιμή του γονιού κόμβου, τότε δε χρειάζεται να μετακινηθεί. Διαφορετικά, ανταλλάσσει θέσεις με τον γονιό του και επαναλαμβάνεται αυτή η διαδικασία μέχρι να βρεθεί στην κατάλληλη θέση. Κατά την εξαγωγή στοιχείου από το min-heap αφαιρείται η ρίζα. Σε αυτήν την περίπτωση, το δεξιότερο στοιχείο στο τελευταίο επίπεδο του δέντρου γίνεται η νέα ρίζα κι έπειτα συγκρίνουμε την τιμή της νέας ρίζας με τους κόμβους-παιδιά της. Σε περίπτωση που η τιμή της ρίζας είναι μικρότερη ή ίση από τις τιμές των παιδιών της, τότε η διαδικασία ολοκληρώνεται εδώ, αλλιώς αντικαθίσταται με τη μικρότερη τιμή κι επαναλαμβάνεται αυτή η διαδικασία μέχρι το δέντρο να διατηρεί τις ιδιότητες ταξινόμησης με τις οποίες υλοποιήθηκε. Αυτό φαίνεται σε παραδείγματα παρακάτω.

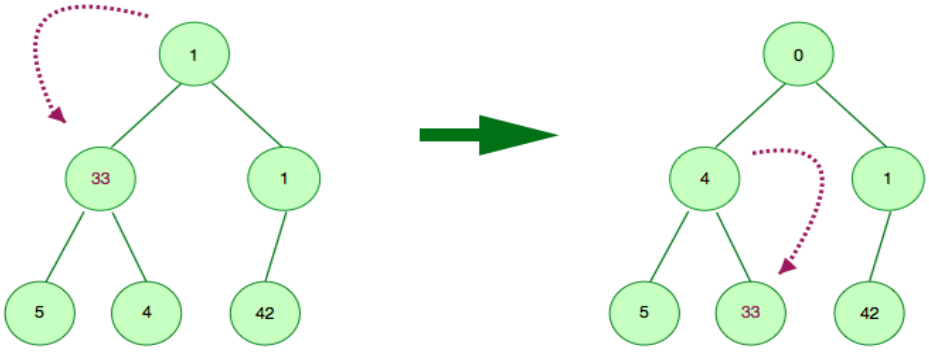
Εισαγωγή του στοιχείου 0





Εξαγωγή στοιχείου





Λεπτομέρειες αλγορίθμου χρονοπρογραμματισμού (scheduling): Κάθε process θα μπαίνει στο min-heap με ένα priority, το οποίο ορίζεται από την εξής σχέση:

$$\frac{runningTime}{currentTime - creationTime}$$

όπου ο αριθμητής είναι ο συνολικός χρόνος που έχει τρέξει η διεργασία και έχει χρησιμοποιήσει τη CPU και ο παρονομαστής είναι ο συνολικός χρόνος που υπάρχει η διεργασία στο σύστημα από την ώρα που δημιουργήθηκε. Συνεπώς, θα χρειαστεί να κρατάτε τις σχετικές πληροφορίες στη δομή του process. Αρχικά, όλα τα νέα processes θα εισάγονται με priority ίσο με 0.

Παράδειγμα: Όπως ξέρουμε, το λειτουργικό εκτελεί ένα RUNNING process έως ότου συμβεί ένα από τα παρακάτω:
- εξαντληθεί το time-slice που του αναλογεί,
- το process σταματήσει σε κάποιο I/O,
- το process τερματίσει.

Όταν το process εξαντλήσει το time-slice και δεν έχει τερματίσει ακόμα, ξαναμπαίνει στη δομή min-heap (δηλαδή γίνεται READY). Τότε ανανεώνεται κατάλληλα το priority του, με βάση τον παραπάνω τύπο, και το process επανατοποθετείται στην κατάλληλη θέση στο min-heap, σύμφωνα με το νέο του priority.

Εκτύπωση αποτελεσμάτων

Στο τελευταίο κομμάτι της άσκησης, θα πρέπει να υλοποιήσετε μια κλήση συστήματος (system call) η οποία θα τυπώνει όλα τα ζωντανά processes στο σύστημα μαζί με κάποιες πληροφορίες για το καθένα. Αυτό σημαίνει ότι θα πρέπει να υλοποιήσετε το system call ώστε να εκτυπώνονται κατάλληλα, όχι μόνο τα ready processes που είναι στο min-heap, αλλά και τα running και τα blocked processes.

Με την κλήση της συνάρτησης που θα υλοποιήσετε θα τυπώνονται για κάθε process:

- Το process ID,
- Το όνομα του,
- Το process ID του γονέα (από ποιό process δημιουργήθηκε),
- Το state του,
- Ο συνολικός χρόνος που έχει εκτελεστεί (runningTime),
- Πόσες φορές έχει γίνει scheduled η διεργασία,
- όποια άλλη πληροφορία θέλετε.

Σύνδεση με Υπάρχουσες Δομές, Δοκιμές

Μελετώντας τον υπάρχοντα scheduler του xv6, μπορείτε να δείτε την υλοποίηση ενός πίνακα που χρησιμοποιείται για την λειτουργία ενός round-robin αλγορίθμου. Για την υλοποίηση σας, τα processes που υπάρχουν σε κάθε θέση του πίνακα αυτού θα

πρέπει να προσπελαύνονται από την δική σας δενδρική δομή. Το τι πληροφορίες θα κρατάτε, και με ποιόν τρόπο, για το σύνολο της δενδρικής δομής αφήνεται στην επιλογή σας. Σε κάθε περίπτωση, θα πρέπει να βρίσκεστε σε θέση να αιτιολογήσετε γιατί επιλέξατε τη συγκεκριμένη υλοποίηση.

Δοκιμάστε το scheduler σας πάνω στο workload που υλοποιήσατε και συγκρίνετε τα αποτελέσματα με τον default scheduler του xv6.

Διαδικαστικά

- Το πρόγραμμά σας θα πρέπει να γραφτεί σε C και να τρέχει στις μηχανές Linux workstations (linuxXY.di.uoa.gr ή linuxvmXY.di.uoa.gr) του τμήματος.
- Οι δομές 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.
- 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 και τις παρατηρήσεις σας.
2. Κατάλληλα τροποποιημένο Makefile του κώδικα του xv6.
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.