Λειτουργικά Συστήματα

(ΕΡΓΑΣΙΑ 1)

ΚΟΛΟΚΥΘΑΣ ΧΡΗΣΤΟΣ

1115201700052

Περιεχόμενα

[1. Εισαγωγή 2](#_Toc90054209)

[2. Έλεγχος παραμέτρων του προγράμματος 2](#_Toc90054210)

[3. Ανάκτηση πληροφοριών για το αρχείο εισόδου 2](#_Toc90054211)

[4. Δημιουργία θυγατρικών διεργασιών 4](#_Toc90054212)

[5. Δέσμευση διαμοιραζόμενης μνήμης και συγχρονισμός μεταξύ διεργασιών 5](#_Toc90054213)

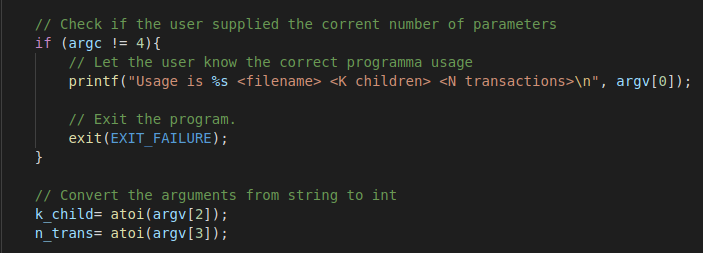
[6. Η λογική των client / server 7](#_Toc90054214)

# Εισαγωγή

Στην παρούσα αναφορά θα αναλυθεί πως αντιμετωπίστηκαν τα επιμέρους ερωτήματα που τέθηκαν στην εργασία 1 του μαθήματος «Λειτουργικά Συστήματα». Σε κάθε επιμέρους τμήμα περιγράφεται το ζητούμενο, παρατίθεται ο κώδικας που αναπτύχθηκε και αναφέρονται τυχόν δυσκολίες και πως αυτές αντιμετωπίστηκαν.

# Έλεγχος παραμέτρων του προγράμματος

Σύμφωνα με την εκφώνηση, η διεργασία δέχεται ως όρισμα ένα αρχείο κειμένου, το πλήθος των διεργασιών παιδιών καθώς και τον αριθμό των δοσοληψιών στις οποίες εμπλέκεται καθεμία εκ των διεργασιών-παιδιών. Έχουμε λοιπόν 4 συνολικά παραμέτρους (πάντα το όνομα του προγράμματος αποτελεί την πρώτη παράμετρο στο πρόγραμμα). Ο έλεγχος για το πλήθος των παραμέτρων είναι απλός και φαίνεται παρακάτω

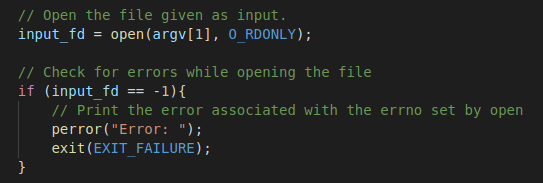


Εάν το πλήθος των παραμέτρων είναι διάφορο του 4, τότε το πρόγραμμα ενημερώνει τον χρήστη για την ορθή χρήση του και τερματίζει. Εάν το πλήθος των παραμέτρων είναι σωστό, τότε μετατρέπονται σε ακεραίους οι αριθμητικές παράμετροι (οι παράμετροι είναι όλοι strings αρχικά) και η εκτέλεση του προγράμματος συνεχίζει.

# Ανάκτηση πληροφοριών για το αρχείο εισόδου

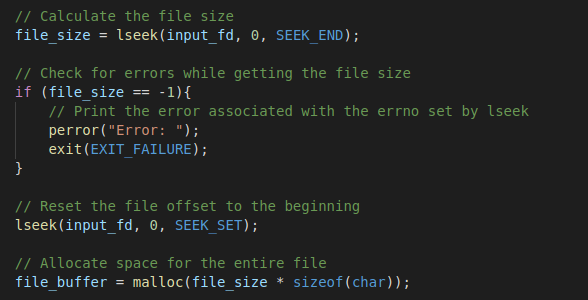
Η γονική διεργασία πρέπει να μετρήσει το πλήθος των γραμμών του αρχείου και να ενημερώσει τις θυγατρικές για αυτό.

Αρχικά, ανοίγουμε το αρχείο που δίνεται ως είσοδος. Αυτό γίνεται με την κλήση συστήματος (syscall) read. Η συνάρτηση αυτή επιστρέφει -1 εάν υπήρξε σφάλμα κατά το άνοιγμα του αρχείου. Ελέγχοντας την τιμή αυτή διακόπτεται η εκτέλεση του προγράμματος εάν το αρχείο δεν βρεθεί για παράδειγμα.

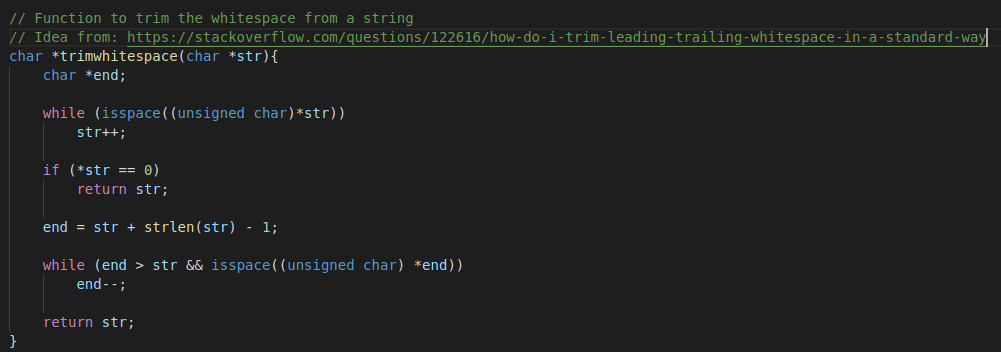


Στο σημείο αυτό έχουμε δύο στόχους. Πρώτον, να μετρήσουμε το πλήθος των γραμμών στο αρχείο εισόδου και δεύτερον να αποθηκεύσουμε το αρχείο στην μνήμη του κυρίως προγράμματος σε τέτοια μορφή η οποία θα διευκολύνει την εξυπηρέτηση των αιτημάτων από τις θυγατρικές διεργασίες. Μια τέτοια δομή είναι ένας char\*\*. Ο δείκτης αυτός δείχνει σε μια περιοχή με δείκτες σε χαρακτήρα. Καθένας από αυτούς τους δείκτες θα δείχνει σε ένα string που θα αντιστοιχεί μία γραμμή του αρχείου. Επιλέχθηκαν δείκτες, καθώς δεν είναι εκ των προτέρων γνωστά ούτε το πλήθος των γραμμών, ούτε το μήκος κάθε μίας.

Πρώτη μας κίνηση λοιπόν, είναι να υπολογίσουμε το μέγεθος του αρχείου. Αυτό γίνεται με χρήση της κλήσης συστήματος lseek. Με την συνάρτηση αυτή, μπορούμε να θέσουν το offset του αρχείου όπου θέλουμε. Η συνάρτηση επιστρέφει κατά πόσα bytes μετακινήθηκε αυτό το offset. Οπότε, εάν το θέσουμε στο τέλος του αρχείου με την επιλογή SEEK\_END, τότε επιστρέφει το μέγεθος του αρχείου σε bytes. Τέλος, πρέπει να επιστρέψουμε το offset του αρχείου στην αρχή του για να συνεχίσουμε με την ανάγνωσή του. Τώρα, γνωρίζοντας το μέγεθος του αρχείου μπορούμε να δεσμεύσουμε μνήμη για αυτό.

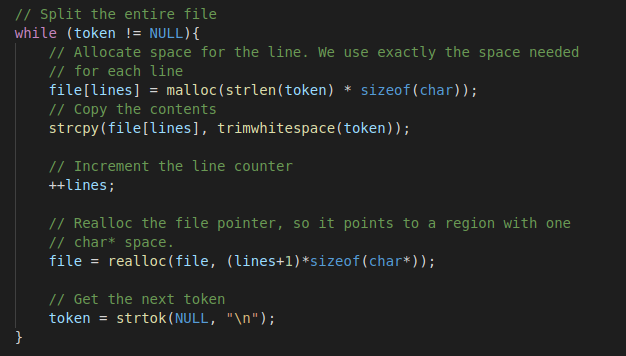


Στη συνέχεια, με χρήση της κλήσης συστήματος read αντιγράφουμε τα δεδομένα από το αρχείο στον buffer file\_buffer. Στον buffer αυτόν, το αρχείο είναι αποθηκευμένο ως μία ενιαία γραμμή. Σκοπός μας τώρα είναι να «σπάσουμε» αυτό το ενιαίο string σε επιμέρους ώστε να πάρουμε τις γραμμές του αρχείου αλλά και το πλήθος τους. Αυτό επιτυγχάνεται με την συνάρτηση strtok(), η οποία διαιρεί ένα string βάσει ενός χαρακτήρα delimiter που ορίζεται από εμάς. Εδώ χρησιμοποιούμε ως delimiter τον χαρακτήρα αλλαγής γραμμής. Σε token που παίρνουμε από την strtok(), αφαιρούμε τα κενά πριν και μετά την γραμμή. Αυτό γίνεται χρησιμοποιώντας την συνάρτηση trimwhitespace().



Η συνάρτηση trimwhitespace - <https://stackoverflow.com/questions/122616/how-do-i-trim-leading-trailing-whitespace-in-a-standard-way>

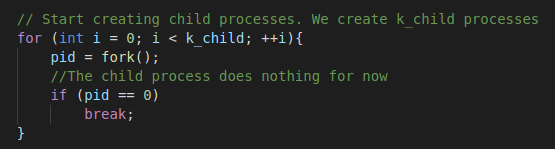
Καθώς παίρνουμε νέες γραμμές, τις αντιγράφουμε στην δομή που θα χρησιμοποιεί το κυρίως πρόγραμμα για να εξυπηρετεί τους πελάτες, μετράμε τις γραμμές που εμφανίζονται και αυξάνουμε τον χώρο που καταλαμβάνει αυτή η δομή καθώς προχωράμε στο αρχείο με χρήση της συνάρτησης realloc().



Στο σημείο αυτό σημειώνεται πως οι μεταβλητές που ορίζονται στο κυρίως πρόγραμμα πριν την κλήση της fork(), αντιγράφονται στις θυγατρικές διεργασίες. Άρα, το πλήθος των γραμμών όπως και εκείνο των δοσοληψιών είναι ήδη γνωστό στις επιμέρους διεργασίες. Επομένως, δεν θα γίνει κάποια έξτρα ενημέρωση από την γονική σε αυτές.

# Δημιουργία θυγατρικών διεργασιών

Η δημιουργία των θυγατρικών διεργασιών γίνεται σε μία απλή επανάληψη for.

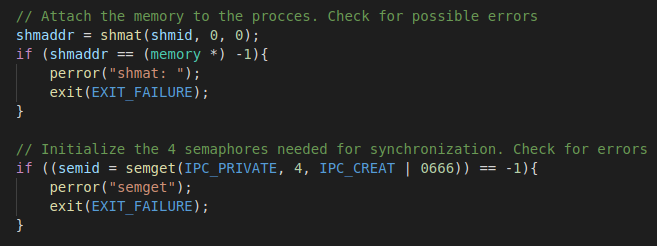


Η fork() επιστρέφει 0 στην διεργασία-παιδί και το pid (process id) στην γονική διεργασία. Στο παραπάνω τμήμα κώδικα φαίνεται πως οι διεργασίες-παιδιά δεν κάνουν τίποτα παραπάνω στην επανάληψη, αντιθέτως βγαίνουν από αυτήν εκτελώντας την εντολή break.

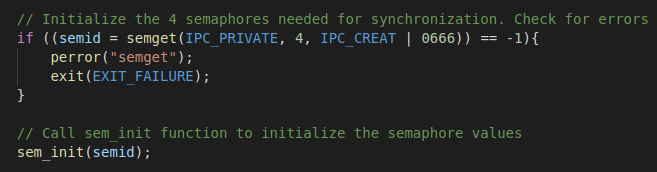
# Δέσμευση διαμοιραζόμενης μνήμης και συγχρονισμός μεταξύ διεργασιών

Για την επικοινωνία μεταξύ των διεργασιών παιδιών και της γονικής, πρέπει να οριστεί μια περιοχή μνήμης οι οποία θα μοιράζεται και από τις δύο. Κάθε διεργασία αποκτά τον δικό της χώρο διευθύνσεων μετά την fork, οπότε η κίνηση αυτή είναι απαραίτητη. Επίσης, για την απρόσκοπτη εκτέλεση το προγράμματος θα πρέπει οι διεργασίες να είναι συγχρονισμένες μεταξύ του, έτσι ώστε να αποφευχθούν περιπτώσεις όπου δύο διεργασίες θα γράφουν ταυτόχρονα στην διαμοιρασμένη μνήμη.

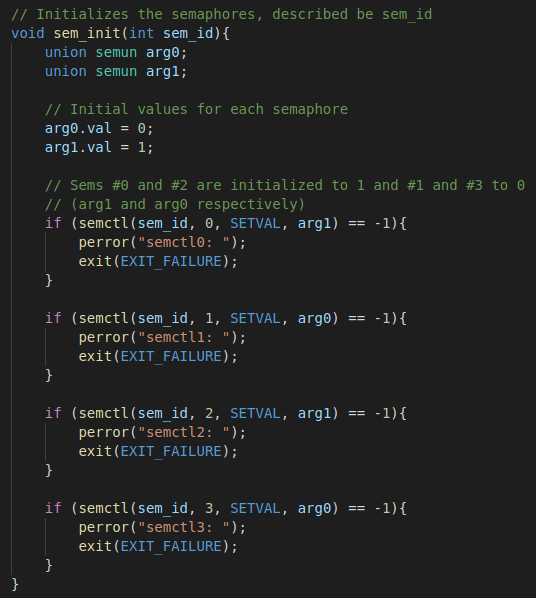
Για την διαμοιρασμένη μνήμη χρησιμοποιείται μια struct, η οποία αποτελείται από δύο buffers. Στον έναν γράφουμε οι θυγατρικές διεργασίες τον αριθμό της γραμμής που ζητούν από την γονική, ενώ στον άλλον γράφει η γονική την γραμμή του αρχείου. Η διαμοιραζόμενη μνήμη δεσμεύεται και προσαρτάται στο πρόγραμμα με τις κλήσεις συστήματος shmget() και shmat().



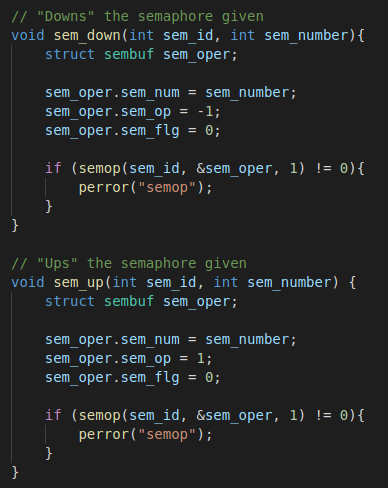
Για τον συγχρονισμό μεταξύ διεργασιών χρησιμοποιούνται σημαφόροι. Αυτοί ορίζονται στο κυρίως πρόγραμμα. Αρχικοποιούνται από την συνάρτηση sem\_init() και ελέγχονται από τις συναρτήσεις sem\_up και sem\_down αντίστοιχα.



Δήλωση 4 σημαφόρων



Αρχικοποίησή τους με χρήση της συνάρτησης sem\_init()



Συναρτήσεις ελέγχου των σημαφόρων

Χρησιμοποιούνται συνολικά 4 σημαφόροι. 2 για κάθε ενέργεια της κάθε οντότητας του προγράμματος (Read/Write και Client/Server).

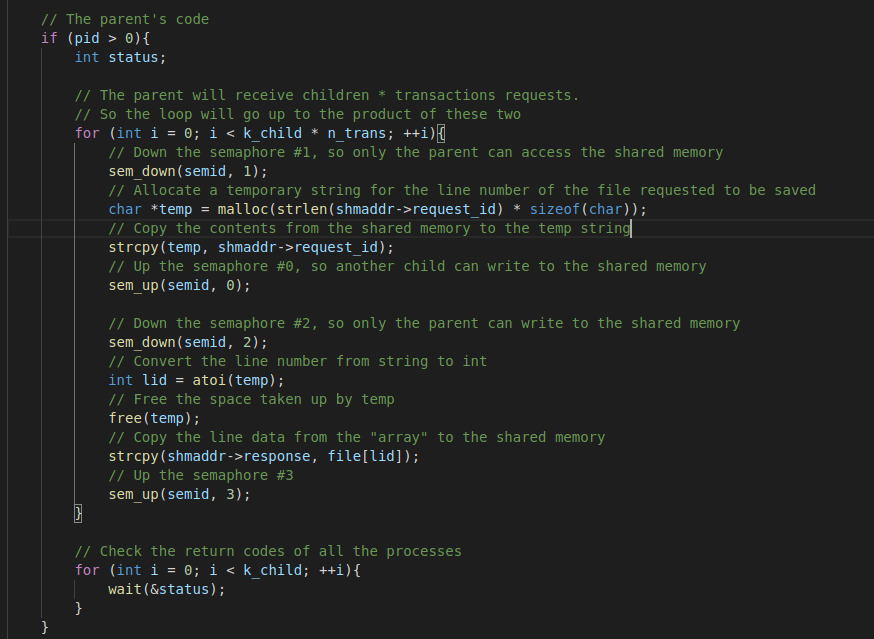
# Η λογική των client / server

Ο γονέας θα δεχτεί συνολικά αιτήματα ίσα με το γινόμενο των παιδιών επί τα αιτήματα που θα στείλει το καθένα. Έτσι, επαναλαμβάνεται η παρακάτω ενέργεια τόσες φορές.

Αρχικά, ο γονέας περιμένει κάποιο παιδί να γράψει στην κοινή περιοχή μνήμης που αντιστοιχεί στα αιτήματα προς αυτόν. Το «περιμένει» επιτυγχάνεται κάνοντας “down” την αντίστοιχη σημαφόρο. Η σημαφόρος αυτή έχει την τιμή 0 άρα η εκτέλεση του γονέα σταματάει μέχρι να γίνει “up” από το παιδί που αποστείλει κάποιο αίτημα. Όταν κάποιο αίτημα φτάσει, τότε ο γονέας συνεχίζει την εκτέλεσή του μετατρέποντας την γραμμή που έστειλε το παιδί από string σε int. Έπειτα, συνεχίζει κάνοντας “up” την σημαφόρο που επιτρέπει σε κάποιο άλλο παιδί να στείλει κάποιο αίτημα.

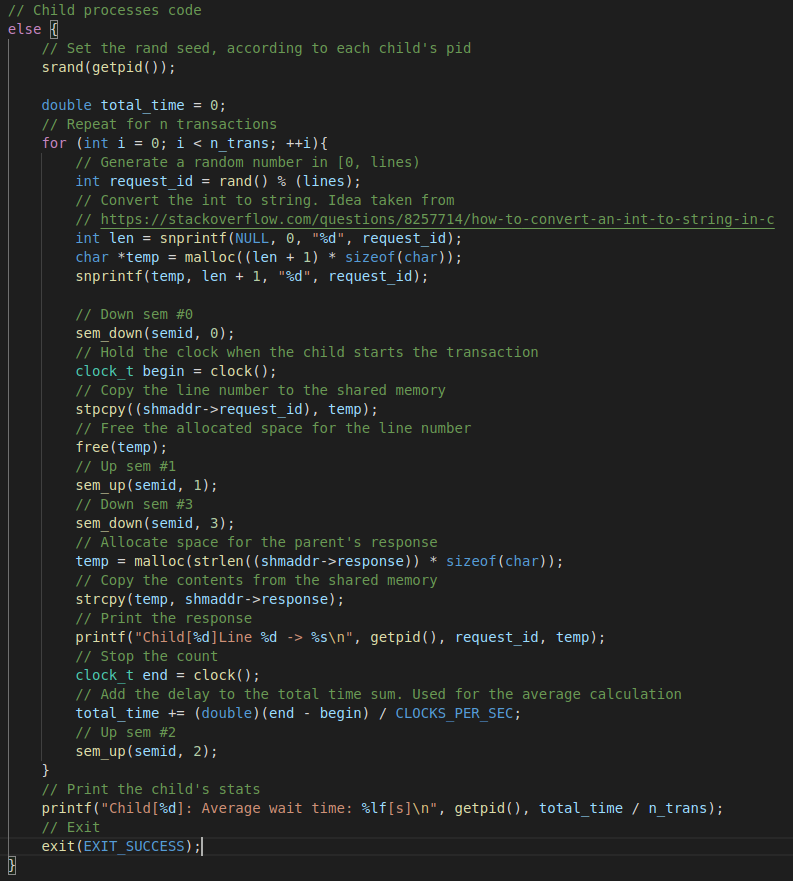
Στη συνέχεια, κάνει “down” την κατάλληλη σημαφόρο ώστε να αποκτήσει αποκλειστική πρόσβαση στην περιοχή μνήμης που γράφονται οι απαντήσεις από αυτόν και γράφει εκεί την ζητούμενη σειρά από το αρχείο.

Τέλος, αφότου ολοκληρωθεί η παραπάνω διαδικασία εξετάζει τις τιμές που επέστρεψαν οι διεργασίες παιδία και τερματίζει.



Κώδικας διεργασίας γονέα

Οι θυγατρικές διεργασίες κάνουν της αντίστοιχη δουλεία, με την διαφορά ότι την επαναλαμβάνουν n\_trans φορές. Οι διεργασίες αυτές παράγουν έναν τυχαίο αριθμό στο διάστημα και τον στέλνουν στην γονική. Περιμένουν (μέσω των σημαφόρων) την απάντηση από την γονική, μετρώντας τον χρόνο που έκανε η γονική να απαντήσει. Ο χρόνος μετράται με χρήση της συνάρτησης clock() και τις σταθεράς CLOCKS\_PER\_CYCLE. Ο χρόνος που εκτυπώνεται στην τυπική έξοδο είναι σε δευτερόλεπτα.



Κώδικας θυγατρικών διεργασιών