**Εικόνα που περιέχει σκίτσο/σχέδιο, καρτούν, εμπρόσθια όψη

Περιγραφή που δημιουργήθηκε αυτόματα**

**ΛΕΙΤΟΥΡΓΙΚΑ ΣΥΣΤΗΜΑΤΑ**

**6Ο εξάμηνο**

**1η εργαστηριακή άσκηση**

**ΠΑΠΑΝΔΡΙΚΟΠΟΥΛΟΣ ΓΡΗΓΟΡΗΣ ΠΑΝΑΓΙΩΤΗΣ ΑΜ 03121136**

**ΝΤΑΒΕΑΣ ΣΤΑΣΙΝΟΣ ΑΜ 03121076**

**Όλοι οι κώδικες των ασκήσεων βρίσκονται ανεβασμένοι στο oslab025**

**ΑΣΚΗΣΗ1(askhsh1.c)**

Εικόνα που περιέχει κείμενο, στιγμιότυπο οθόνης, λογισμικό, λογισμικό πολυμέσων

Περιγραφή που δημιουργήθηκε αυτόματα



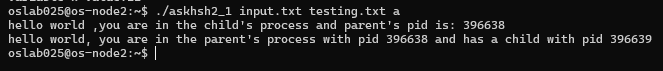


Στον παραπάνω κώδικα υλοποιήσαμε την άσκηση 1 (αναγνώριση αρχείων στη C με την βοήθεια κλήσεων συστήματος.).Η main παίρνει δύο ορίσματα το ένα ελέγχει πόσα ορίσματα έχουμε βάλει κατά την εκτέλεση και το δεύτερο τα ορίσματα αυτά κάθε αυτά τα οποία είναι τα .txt αρχεία στα οποία από το ένα διαβάζουμε και στο άλλο γράφουμε το πόσες φορές εμφανίζεται το γράμμα που έχουμε ορίσει ως τελευταίο όρισμα argv[3][0].Αρχικά κάνουμε έναν έλεγχο αν έχει δοθεί ο κατάλληλος αριθμός ορισμάτων κατά την εκτέλεση του προγράμματος, αν είναι διάφορος του 4 τότε μέσω της exit το πρόγραμμα τερματίζει ενώ παράλληλα μέσω της printf τυπώνουμε στο cmd ότι o αριθμός των ορισμάτων είναι λάθος.Έπειτα ορίζουμε τους περιγραφητές αρχείων fdr kαι fdw για τα αρχεία ανάγνωσης και εγγραφής αντίστοιχα. Ενώ στην συνέχεια με την fopen ανοίγουμε το αρχείο argv[1] το οποίο και ανοίγουμε για διάβασμα όπως υποδεικνύει η O\_RONLY ενώ παράλληλα ελέγχουμε και αν επιστρέφει -1.Σε αυτή την περίπτωση έχει συμβεί λάθος κατά την κλήση της και για αυτό τερματίζουμε το πρόγραμμα μέσω της exit και τυπώνουμε λάθος στο cmd μέσω της perror .Αντίστοιχα δουλεύουμε και για το αρχείο που θα γράψουμε argv[2] mε μόνη διαφορά ότι θα βάλουμε στην open O\_WRONLY | O\_CREAT καθώς γράφουμε και δημιουργούμε το αρχείο .Επειτα, χρησιμοποιούμε μία while loop η οποία τερματίζει μέχρι να σταματήσει η read να διαβάζει το πρόγραμμα. Η read ισούται με μηδέν όταν τελειώσει το διάβασμα του αρχείου. Επίσης έχει όρισμα τον περιγραφητή του αρχείου που διαβάζουμε fdr έπειτα, δυναμικά το cc που εχουμέ ορίσει ως char kαι τέλος το 1 γιατί το cc είναι μεγέθους 1 και μέσα στην loop κάθε φορά που το argv[3][0] ισούται με το cc δηλαδή τον χαρακτήρα στον οποίο βρίσκεται η read αν είναι ίσα ο counter (count) αυξάνεται κατά 1.Γενικά η read επιστρέφει το πλήθος των bytes που διάβασε. Έπειτα ,μέσω της dprintf όπου παίρνει ως όρισμα των περιγραφητη αρχείου στον οποίο θα γράψουμε τυπώνει σε αυτό πόσες φορές βρέθηκε στο άλλο αρχείο ο χαρακτήρας που αναζητήσαμε. Εναλλακτικά μπορούσαμε να χρησιμοποιήσουμε την εντολή write θα αρχικοποιούσαμε στατικά ένα πίνακα buf αρκετά μεγάλο,Έπειτα μέσω της sprintf θα τυπώναμε το επιθυμητό μήνυμα και μέσω αυτής θα επιστρεφόταν το μήκος σε bytes της συμβολοσειράς που πρέπει να τυπώσουμε. Τέλος, μέσω της write η οποία παίρνει ως ορίσματα τον fdw τo len και το buf τυπώνουμε το επιθυμητό μήνυμα στο αρχείο ενώ παράλληλα ελέγχουμε αν έχει συμβεί σφάλμα κατά την εκτέλεσή του,αν δηλαδή επιστρέφει -1.Τέλος μέσω της εντολής close κλείνουμε τα αρχεία στα οποία δουλέψαμε.

**ΑΣΚΗΣΗ2.1(askhsh2\_1.c)**

Εικόνα που περιέχει κείμενο, στιγμιότυπο οθόνης, λογισμικό, οθόνη

Περιγραφή που δημιουργήθηκε αυτόματα



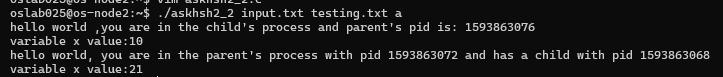


Στο πρώτο ερώτημα της άσκησης 2 μέσω της εντολής fork() δημιουργούμε ένα αντίγραφο του πατέρα δηλαδή ένα παιδί που κάνει ακριβώς τις ίδιες διεργασίες με τον πατέρα με μόνη διαφορά ότι έχει διαφορετικό αναγνωριστικό (pid) .Κάνουμε τους ακόλουθους ελέγχους. Εάν ,το p που επιστρέφει η fork ισούται με -1 η εκτέλεση της fork απέτυχε και μέσω της perror στο cmd τυπώνεται στο πρόγραμμα ότι η fork failed ενώ μέσω της exit το πρόγραμμα τερματίζει. Ένα η fork επιστρέφει 0 τότε το πρόγραμμα εκτελεί την διεργασία παιδί. Μέσω της κλήσης της getppid επιστρέφεται το process id του πατέρα του ,μέσω της getpid το process id του παιδιού ,τα τυπώνουμε στην έξοδο ,και με την exit τερματίζουμε το child process. Τέλος στην περίπτωση που p>0 βρισκόμαστε στον πατέρα μέσω της wait επιστρέφουμε το pid του παιδιού που δημιούργησε και μέσω του getpid το process id του.Έπειτα, τα τυπώνουμε και μέσω της exit τερματίζουμε το parent process

**AΣΚΗΣΗ2.2(askhsh2\_2.c)**

Εικόνα που περιέχει κείμενο, στιγμιότυπο οθόνης, λογισμικό, λογισμικό πολυμέσων

Περιγραφή που δημιουργήθηκε αυτόματα





Γενικά, κάθε διεργασία έχει το δικό της εικονικό χώρο διευθύνσεων μνήμης, ο οποίος μπορεί να αντιστοιχιστεί σε διαφορετικές φυσικές σελίδες μνήμης. Ο εικονικός χώρος διευθύνσεων αντιπροσωπεύει αυτό που μια διεργασία βλέπει ως διευθύνσεις μνήμης. Από την άλλη πλευρά, ο φυσικός χώρος διευθύνσεων αναφέρεται στην πραγματική μνήμη RAM που χρησιμοποιείται από το σύστημα. Αυτή είναι η πραγματική φυσική μνήμη που αποτελεί το περιβάλλον εκτέλεσης μιας διεργασίας. Κατά την εκτέλεση, το λειτουργικό σύστημα μεταφράζει τις εικονικές διευθύνσεις που χρησιμοποιεί η διεργασία σε φυσικές διευθύνσεις στην πραγματική μνήμη, αντιστοιχίζοντας τις εικονικές διευθύνσεις σε φυσικές σελίδες μνήμης.

Στο πατέρα ορίζουμε την μεταβλητή x και της αναθέτουμε την τιμή 21.Στο παιδί αλλάζουμε αυτή την τιμή (x = 10).Η fork αντιγράφει στο παιδί την διεργασία του πατέρα. Ότι συμβαίνει στον παιδί όμως δεν κληρονομείται στον πατέρα με αποτέλεσμα όταν τυπώνουμε το x στον πατέρα και στο παιδί να τυπώνονται διαφορετικά αποτελέσματα

**AΣΚΗΣΗ2.3(askhsh2\_3.c)**

Εικόνα που περιέχει κείμενο, στιγμιότυπο οθόνης, λογισμικό, οθόνη

Περιγραφή που δημιουργήθηκε αυτόματαΕικόνα που περιέχει κείμενο, στιγμιότυπο οθόνης, γραμματοσειρά, μαύρο

Περιγραφή που δημιουργήθηκε αυτόματαΑυτό το οποίο κάναμε στην άσκηση 2.3 είναι να μεταφέρουμε τις διεργασίες της 2.1 στον πατέρα και ο έλεγχος χαρακτήρων να υλοποιείται από το παιδί και να τυπώνεται σε αυτό το περιεχόμενο του αρχείου εγγραφής.

**AΣΚΗΣΗ2\_4(askhsh2\_4.c)**

Εικόνα που περιέχει στιγμιότυπο οθόνης, κείμενο, λογισμικό, λογισμικό πολυμέσων

Περιγραφή που δημιουργήθηκε αυτόματα

Στο πρόγραμμα αυτό όταν βρισκόμαστε στην διεργασία παιδί δημιουργούμε ένα πίνακα args[] που παίρνει σαν όρισμα το executable του δοθέντος αρχείου και τα υπόλοιπά τρία ορίσματα που απαιτεί για την ορθή εκτέλεσή του. Έπειτα καλούμε την συνάρτηση execv η οποία παίρνει σαν όρισμα την args[0],args και αυτό το οποίο κάνει κατά την εκτέλεσή της είναι να επιτρέπει στην διεργασία παιδί να εκτελέσει το πρόγραμμα που μας είχε δοθεί.

**ΑΣΚΗΣΗ3(askhsh3.c)**

Εικόνα που περιέχει κείμενο, ηλεκτρονικές συσκευές, στιγμιότυπο οθόνης, λογισμικό

Περιγραφή που δημιουργήθηκε αυτόματα

Εικόνα που περιέχει κείμενο, στιγμιότυπο οθόνης, λογισμικό

Περιγραφή που δημιουργήθηκε αυτόματα

Εικόνα που περιέχει κείμενο, γραμματοσειρά, στιγμιότυπο οθόνης, μαύρο

Περιγραφή που δημιουργήθηκε αυτόματα

Στην άσκηση 3 δημιουργήσαμε P διεργασίες παιδιά όπου κάθε ένα από αυτά διάβάζει ένα συγκεκριμένο κομμάτι του αρχείου εγγραφης. Συγκεκριμένα, για να υλοποιήσουμε αυτό τον σκοπό κάναμε τις ακόλουθες ενέργειες. Αρχικά κάναμε με #define τον αριθμό των παιδιών που δημιουργήσαμε μέσω fork kαι ενός for loop. Ακόμη ,δημιουργήσαμε pipes μέσω for loop ένα για κάθε παιδί ώστε να επικοινωνεί το καθένα μέσω διοχέτευσης με τον πατέρα. Ορίσαμε έναν δισδιάστατο πίνακα P\*2 file descriptors .Συγκεκριμένα κάθε σωλήνωση πατέρα-παιδιού θέλει ένα file descriptor για εγγραφή και έναν file descriptor για ανάγνωση pipefd[i][0] και pipefd[i][1] για το άκρο ανάγνωσης και εγγραφής αντίστοιχα .Κάθε διοχέτευση δημιουργήθηκε μέσω της κλήσης του συστήματος pipe που παίρνει σαν όρισμα το πίνακα του filedescriptor. Ακόμη προκειμένου να διαβάζει κάθε παιδί ένα συγκεκριμένο κομμάτι του αρχείου ανάγνωσης χρησιμοποιήσαμε τις συναρτήσεις fstat kαι lseek. Μέσω της fstat παίρνουμε το μέγεθος σε bytes του αρχείου και το διαιρούμε διά τον αριθμό των παιδιών προκειμένου κάθε παιδί να διαβάζει ένα συγκεκριμένο κομμάτι του αρχείου και ίσο σε μέγεθος με τα υπόλοιπα παιδιά. Έπειτα για κάθε παιδί οριοθετούμε από που μέχρι που θα διαβάσει το αρχείο start end Μέσω της lseek ορίζουμε που θα ψάξει. Την καλούμε κάθε φορά στο loop βάζοντας της κάθε φορά σαν όρισμα τον περιγραφητή αρχείου του παιδιού την αρχή που για κάθε παιδί αλλάζει και μέσω τις ετικέτας SEEK\_SET μετακινούμε τον δείκτη κατά start bytes από την αρχή του αρχείου.Ακόμη στο while loop βάζουμε ως επιπλέον προυπόθεση το lseek<end δηλαδή να διαβάσει μέχρι εκεί που του έχουμε οριοθετήσει. Μετά, γράφουμε στο άκρο εγγραφής του pipe πόσους χαρακτήρες βρήκαμε και έπειτα το κλείνουμε. Στον πατέρα ανοίγουμε το άκρο ανάγνωσης ενώ παράλληλα κλείνουμε το άκρο εγγραφής και διαβάζουμε πόσους χαρακτήρες βρήκαμε και τους προσθέτουμε στον total\_counter. Μέσω της dprintf τυπώνουμε το total\_count στο αρχείο εγγραφής ,ενώ έχουμε και ένα for loop που καλεί την wait προκειμένου ο πατέρας να αναμένει πρώτα το τερματισμό του κάθε παιδιού .Όσον αφορά τα signals kαι την διαχείριση του crl ^c κάναμε τα ακόλουθα.Δημιουργήσαμε ένα handler που όταν καλείται αντί να τερματίζει τυπώνει πόσα παιδιά έχουμε δημιουργήσει. Την handler την καλούμε στην main μέσω της signal που παίρνει ως όρισμα το handler και το SIGINT δηλαδή την εντολή που θα παρακάμψουμε Η μόνη διαφορά της με την sigaction είναι ότι δεν προτιμάται για λόγους φορητότητας. Ωστόσο λόγω ευκολίας την επιλέξαμε. Τέλος καλέσαμε την sleep σε μία λούπα προκειμένου να υπάρχει χρόνος αντίδρασης στην εκτέλεση του signal.