ΛΕΙΤΟΥΡΓΙΚΑ ΣΥΣΤΗΜΑΤΑ

ΠΡΩΤΗ ΕΡΓΑΣΙΑ

ΠΕΡΙΕΧΟΜΕΝΟ ΕΡΓΑΣΙΑΣ : ΔΗΜΙΟΥΡΓΙΑ ΕΝΟΣ ΚΕΛΥΦΟΥΣ (SHELL) ΣΤΟ ΛΕΙΤΟΥΡΓΙΚΟ ΣΥΣΤΗΜΑ ΤΟΥ LINUX.

ΣΥΜΜΕΤΕΧΟΝΤΕΣ :

1. ΦΑΣΟΥΛΑΣ ΘΕΟΔΩΡΟΣ
   (ΑΕΜ:2096)
2. ΓΕΩΡΓΙΑΔΗΣ ΝΙΚΟΛΑΟΣ
   (ΑΕΜ :2043)
3. ΚΟΥΝΤΟΥΡΙΩΤΗΣ ΣΩΤΗΡΙΟΣ
   (ΑΕΜ:2108)
4. ΑΛΕΞΑΝΔΡΟΠΟΥΛΟΣ ΕΥΘΥΜΙΟΣ
   (ΑΕΜ:2113)

- ΕΙΣΑΓΩΓΗ

Για την πραγματοποίηση αυτής της εργασίας χρησιμοποιήθηκε η έκδοση VirtualBox 4.2 (για την προσομοίωση) και η Ubuntu 12.04 έκδοση του λειτουργικού συστήματος Linux.

Θα αναλύσουμε ξεχωριστά κάθε τμήμα του κώδικα που αφορά τις συναρτήσεις και στη συνέχεια θα τα συνδέσουμε για να εξηγήσουμε το πώς λειτουργεί η main.

- Parsing(char *cmd,char *buffer)

  {

  Στην εργασία μας χρησιμοποιήθηκαν συναρτήσεις στις οποίες ανατέθηκε μία συγκεκριμένη λειτουργία του προγράμματος .Μία από αυτές είναι και η μέθοδος parsing(…) η οποία δέχεται ως ορίσματα ,την συμβολοσειρα cmd στην οποία αποθηκεύεται η εντολή που δέχεται το προγραμμα μας από τον χρήστη και την συμβολοσειρά buffer που θα διευκρινιστεί στην συνέχεια.

  Αυτό που κανει το parsing είναι να παίρνει κάθε φορά την εντολη του χρηστη να την σπάει σε κομμάτια και να ορίζει τον τύπο και τον τόπο της εντόλης που θα εκτελεστεί.

  Αρχικά οριζουμε και δεσμευουμε χωρο για ενα δυσδιαστατο πίνακα **tmp 5 θέσεων, το μεγεθος του οποίου μας το καθορίζουν τα 4 ορισματα το πολύ.Η λειτουργία του πίνακα αυτού είναι αρκετά χρησιμη για τη διεξαγωγή του parsing αφου αποθηκεύει σε κάθε θέση του κομματια της εντολής η οποία έχει κομματιαστεί χωρις τα κενά.Η μέθοδος που χρησιμοποιούμε για να πραγματοποιήσουμε αυτό το σπάσιμο,καλείται strtok και παιρνει ως ορίσματα την εντολη μας και ενα πινακα με χαρακτηρες για τους οποιους πραγματοποιειται το σπασιμο εφοσον συναντησει τουλαχιστον εναν απο αυτους.Ετσι εχουμε την strtok(cmd,” \t\n”); και τωρα σε περιπτωση που συναντησει κενο θα σπασει την cmd σε δύο συμβολοσειρες. Επειδη εμεις θελουμε να κρατησουμε την εντολη που θα εκτελεστει χρησιμοποιουμε εναν βοηθητικο πινακα τον char *cm2 .

Ο πινακας cmd2 ειναι αυτος στον οποιο επιστρεφεται καθε φορα το αποτελεσμα της strtok(..)..

- cd

Το shell μας υποστηριζει διαφορες εντολες .Μια απο αυτες ειναι και η change directorty η αλλιως cd.Για να την υλοποιησουμε δημιουργουμε μια επαναληψη η οποία εκτελειται οσο η cmd2 δεν ειναι null. Αυτο με αλλα λογια μεταφραζεται σε «οσο η strtok εχει συμβολοσειρα να τεμαχισει>> .Στη συνεχεια κανουμε εναν ελεγχο για το αν οντως η εντολη του χρηστη ειναι η cd με την strcmp και χρησιμοποιουμε παλι την strtok για να παρουμε το μονοπατι στο οποιο επιθυμουμε να μεταβουμε.Ετσι τωρα εχοντας το μονοπατι χρησιμοποιουμε την κληση συστηματος chdir που επιστρεφει ‘0’ αν το μονοπατι υπαρχει και μπορουμε να μεταβουμε σε αυτό.Σε αντιθετη περιπτωση εμφανιζει μηνυμα λάθους.

- pwd

Αντιστοιχα το shell μας υποστηριζει την pwd η οποια δεν κανει τιποτα αλλο απο το να επιστρεφει το μονοπατι στο οποιο βρισκομαστε εκεινη την χρονικη στιγμη που την καλουμε.Χρησιμοποιειται η γνωστη κληση συστηματος getcwd και ενα βοηθητικο buffer για να το τυπωνουμε στην οθόνη.

- ls,cat,ws

Στη συνεχεια για αυτες τις 3 εντολες θα χρησιμοποιηθει η γνωστη κληση συστηματος execvp(…) η οποια χρησιμοποιει τον βοηθητικο πινακα tmp 5 θεσεων ο οποιος θελουμε να εχει στην καθε θεση του διαφορετικο κομματι εντολης. Για παραδειγμα στη θεση μηδεν θελουμε να υπαρχει το ονομα της

εντολης και στις υπολοιπες τα ορισματα της ,οπως για παραδειγμα το path που θα εκτελεστει αυτη. Π.χ tmp[0]=”ls”

tmp[1]=”/bin/usr/Desktop”

Για να το κανουμε αυτο δημιουργουμε μια επαναληψη for και μεσα στο σωμα της , αναθετουμε σε καθε θεση του tmp το αποτελεσμα της strtok η οποια παιρνει ως πρωτο ορισμα null.

Στη συνεχεια αφου τελειωσει η for καλουμε την κληση συστηματος execvp με πρωτο ορισμα το tmp[0] που ειναι η τρεχουσα εντολη και δευτερο ορισμα τον πινακα tmp.Επειτα καλειται η break; για να περασουμε στο επομενο σταδιο της επαναληψης αν αυτο υπαρχει.

- MAIN()

  {

  Στη main οριζουμε αρχικα τα απαραιτητα arrays που αναφερθηκαν προηγουμενως οπως (cmd ,buffer) και δεσμευουμε τον απαραιτητο χωρο για αυτα.Στη συνεχεια δημιουργουμε ενα loop while(1) το οποιο εκτελειται για οσο ο χρηστης εχει ανοιχτο το shell και πληκτρολογει οσες εντολες επιθυμει μεχρις οτου πληκτρολογησει exit για να τερματισει την ολη διαδικασια.

  Αρχικα τυπωνεται η καταλληλη προτροπη (MyShell$) μαζι με το τρεχων path που το θεωρησαμε αρκετα χρησιμο για τον χρηστη.Στη συνεχεια ορίζεται μια μεταβλητη id τυπου pid_t ο οποιος αντιπροσωπευει το id της διεργασιας που θελουμε να δημιουργησουμε και καλουμε τη fork();.

Αυτη με τη σειρα της θα δωσει εναν αριθμο σε αυτο id ο οποιος θα παρει καμια τιμη αρνητικη,θετικη,μηδεν.
Διακρινουμε τρεις περιπτωσεις.Σε περιπτωση που το id παρει τιμη αρνητικη η fork() απετυχε να δημιουργησει την διεργασια-Failed .Εαν το id παρει τιμη ιση με το μηδεν τοτε η fork() εχει δημιουργησει μια διεργασια παιδι-Child Process.Τελος αν αυτη η τιμη ειναι θετικη τοτε εχουμε να κανουμε με μια θυγατρικη διεργασια – Parent Process.

Τωρα θα αναλυσουμε τι γινεται σε καθε περιπτωση .
-Failed.

Εαν το id ειναι αρνητικο τοτε αποτυχαμε και τυπωνουμε το αντιστοιχο μηνυμα στην οθονη.

-Child Process

Εαν εχουμε να κανουμε με Child τοτε καλουμε την μεθοδο parsing ,αποκωδικοποιει την εντολη με τον τροπο που περιγραψαμε πιο πανω και εκτελουνται οι απαραιτητες ενεργειες.

-Parent Process

Στην περιπτωση που εχουμε Parent τοτε καλειται η wait η οποια περιμενει την Θυγατρικη διεργασια που λειτουργει στο προσκηνιο να τελειωσει.Αντιθετα αν εχουμε παρασκηνιο βρισκουμε το ampersand στο τελος το σβηνουμε και στο σημειο αυτο η Parent θα βγει απο το loop ,δηλαδη δεν θα περιμενει καθολου.Ο χρηστης ειναι τωρα ετοιμος να πληκτρολογησει την επομενη εντολή.