ΠΑΝΕΠΙΣΤΗΜΙΟ ΚΥΠΡΟΥ
Τμήμα Πληροφορικής
ΕΠΛ 371 - Προγραμματισμός Συστημάτων

ΑΣΚΗΣΗ 3 – Υλοποίηση Κελύφους UCYSH με Κλήσεις Συστήματος

Διδάσκων: Δημήτρης Ζεϊναλιπούρ
Υπεύθυνος Εργαστηρίου: Παύλος Αντωνίου

Ημερομηνία Ανάθεσης: Παρασκευή 18/03/2016
Ημερομηνία Παράδοσης: Πέμπτη 31/03/2016 (13 μέρες)

(να παραδοθεί ο εκτυπωμένος κώδικας στο εργαστήριο και να υποβληθεί ηλεκτρονικά στο Moodle)

I. Στόχος Άσκησης

Στόχος αυτής της άσκησης είναι η εξοικείωση με προχωρημένες τεχνικές προγραμματισμού στο περιβάλλον UNIX. Επίσης θα εξοικειωθείτε με κλήσεις συστήματος όπως τη fork() για τη δημιουργία διεργασιών και τη χρήση σημάτων όπως για παράδειγμα την αναμονή τερματισμού μιας διεργασίας – παιδί. Η άσκηση περιλαμβάνει την υλοποίηση του δικού σας κελύφους (ucy shell – ucysh) στη γλώσσα C. Αυτό το κέλυφος θα συμπεριφέρεται παρόμοια με τα κοινά shell, ειδικότερα το bash, με περιορισμένες λειτουργίες (δεν θα υποστηρίζει δομές προγραμματισμού και προχωρημένες λειτουργίες. Οι λειτουργίες του προγράμματος σας και το αναμενόμενο αποτέλεσμα περιγράφονται αναλυτικότερα στην συνέχεια.

II. Περιγραφή των ζητουμένων λειτουργιών του ucysh

Γενική Περιγραφή

Η σύνταξη των εντολών του κελύφους σας θα πρέπει να ακολουθεί τον τρόπο με τον οποίο καλούνται στο κέλυφος bash. Μπορείτε να δείτε όλες τις πληροφορίες πληκτρολογώντας την εντολή: `man bash`
Το κέλυφος σας, ucysh, θα πρέπει να παρουσιάζει αποτελέσματα χρησιμοποιώντας:

i. βασικές εντολές του bash shell με ορίσματα (π.χ. ls, ls -a, wc) με χρήση exec(),
ii. καθολικές και τοπικές μεταβλητές κελύφους, και
iii. built-in συναρτήσεις.

Σε σχέση με τις μεταβλητές, το κέλυφος σας θα πρέπει να κληρονομεί τις καθολικές μεταβλητές του κελύφους από το οποίο ξεκίνησε και ταυτόχρονα θα μπορεί να κρατά τις δικές του τοπικές μεταβλητές που ορίζει ο χρήστης. Θα πρέπει επίσης να υποστηρίζονται δυναμικές μεταβλητές τύπου $RANDOM, $HOSTNAME, κτλ. Σε σχέση με τις καθολικές μεταβλητές, ο χρήστης του κελύφους σας θα πρέπει να είναι σε θέση να δει και να αλλάξει <u>οποιαδήποτε</u> από τις υπάρχουσες μεταβλητές αλλά και να προσθέσει δικές του (δες παράγραφο IV πιο κάτω).

Επιπλέον, το κέλυφος σας πρέπει να καλύπτει τις builtin συναρτήσεις του bash που αναφέρονται στις διαφάνειες 8-29 και 8-30 χρησιμοποιώντας standard I/O library calls ή system calls. Για παράδειγμα, η εντολή `read -p "Give a number" a` θα υλοποιηθεί με χρήση της εντολής printf και scanf.

Λειτουργίες

Το δικό σας κέλυφος θα πρέπει να υποστηρίζει τις πιο κάτω λειτουργίες:

- Εκτέλεση βασικών εντολών με ορίσματα γραμμής εντολών (command line arguments).
- Δυνατότητα εξόδου από το κέλυφος.
- Να επιτρέπεται η εκτέλεση εντολών τόσο στο προσκήνιο (foreground) όσο και στο παρασκήνιο (background) χρησιμοποιώντας το σύμβολο &. Οι εντολές που δεν θα τερματίζονται με το σύμβολο & θα τρέχουν στο προσκήνιο ενώ οι εντολές με το & θα τρέχουν στο παρασκήνιο.
- Να επιτρέπεται η ανακατεύθυνση εισόδου/εξόδου (I/O redirection). Το σύμβολο < θα χρησιμοποιείται για την ανακατεύθυνση του standard input από αρχείο αντί από το πληκτρολόγιο, και το > για την ανακατεύθυνση του standard output σε αρχείο αντί στην οθόνη.
- Το σύμβολο `;` να διαχωρίζει δύο ξεχωριστές εντολές αλλά να τους επιτρέπει να δίνονται από το χρήστη στην ίδια γραμμή, όπως π.χ. `ls;cd ..`, οι οποίες πρώτα θα παρουσιάζουν τα περιεχόμενα του τρέχοντος καταλόγου και μετά θα μεταφέρεται ο χρήστης στον γονικό κατάλογο.
- Να υποστηρίζεται η εντολή pipe, `|`. Για παράδειγμα η γραμμή `ls –l | more` παίρνει την έξοδο της εντολής `ls –l` και την χρησιμοποιεί σαν είσοδο στην εντολή `more`. Στην εντολή αυτή δεν πρέπει να εμπλέκεται εγγραφή/ανάγνωση σε/από αρχεία.
- Το κέλυφος θα κρατά ένα μετρητή του συνολικού αριθμού των διεργασιών που έγιναν fork και να τον τυπώνει στην αρχή κάθε γραμμής του κελύφους.
- Το κέλυφος θα κρατά ένα μετρητή για τον αριθμό των τρεχουσών ζωντανών διεργασιών. Αν υπάρχουν πάνω από 10 τρέχουσες ζωντανές διεργασίες το κέλυφος δεν θα αφήνει να ξεκινήσουν άλλες, αλλά να τυπώνεται το μήνυμα "Insufficient Resources" έτσι ώστε ο χρήστης θα προσπαθήσει ξανά αργότερα. Η εντολή (μπορεί να είναι σύνολο εντολών) που προκάλεσαν το πρόβλημα αυτό θα ακυρώνεται.
- Για κάθε εντολή, το κέλυφος θα κάνει fork μια νέα διεργασία (process).

III. Παραδείγματα εκτέλεσης εντολής

Πιο κάτω δίνεται ένα παράδειγμα εκτέλεσης με μερικές από τις εντολές που ζητούνται προς υλοποίηση.

```
b103ws10:/home/faculty/csp5pa1>./ucysh
0-ucysh>ls
a.out ucysh.c parse.c
1-ucysh>./a.out
0-ucysh>exit
2-ucysh>exit
b103ws10:/home/faculty/csp5pa1>./ucysh
0-ucysh>ls > temp
1-ucysh>more temp
a.out
ysh.c
parse.c
temp
2-ucysh>wc < temp
4 4 25
3-ucysh>rm temp
4-ucysh>exit
b103ws10:/home/faculty/csp5pa1>
```

Μέσα στο κέλυφος ucysh μπορείτε επίσης να υλοποιήσετε και οτιδήποτε άλλο θέλετε (που δεν αναφέρεται πιο πάνω) για βελτίωση του βαθμού σας.

VI. Βοηθητικές Συναρτήσεις και Μεταβλητές

Για την διεκπεραίωση αυτής της άσκησης πιθανόν θα χρειαστείτε κάποιες συναρτήσεις και μεταβλητές τις οποίες δεν είδαμε στο μάθημα όπως:

```
#include <stdlib.h>
Function: char *getenv(const char *name);
This function returns a string that is the value of the environment
variable name. You must not modify this string. If the environment
variable name is not defined, the value is a null pointer.

Function: int putenv (char *string);
This function adds or removes definitions from the environment. If
the string is of the form 'name=value', the definition is added to the
environment. Otherwise, the string is interpreted as the name of an
environment variable, and any definition for this variable in the environment
is removed. If the function is successful it returns 0. Otherwise the return
value is nonzero and errno is set to indicate the error.

Function: int setenv (const char *name, const char *value, int replace);
This function can be used to add a new definition to the environment. The
entry with the name name is replaced by the value 'name=value'. Please note
that this is also true if value is the empty string. To do this a new string
is created and the strings name and value are copied. A null pointer for
the value parameter is illegal. If the environment already contains an entry
with key name the replace parameter controls the action. If replace is zero,
nothing happens. Otherwise the old entry is replaced by the new one.

Variable: char ** environ
The environment is represented as an array of strings. Each string is of the
format 'name=value'. The order in which strings appear in the environment is
not significant, but the same name must not appear more than once. The last
element of the array is a null pointer.
This variable is declared in the header file unistd.h.
If you just want to get the value of an environment variable, use getenv.
Unix systems, and GNU systems, pass the initial value of environ as the third
argument to main.
```

Μπορείτε να δείτε περισσότερα εδώ:
http://www.gnu.org/software/libc/manual/html_node/Environment-Access.html#Environment-Access

IV. Γενικοί Κανόνες

1. Το πρόγραμμα σας πρέπει να χρησιμοποιεί τεχνικές δομημένου προγραμματισμού με την χρήση συναρτήσεων, πολλαπλών αρχείων για καλύτερη δομή του πηγαίου κώδικα, Makefile, διαχείριση λαθών συστήματος με την perror κτλ.;
2. Το πρόγραμμα σας πρέπει να ελαχιστοποιεί την χρήση πόρων του συστήματος (μνήμης, επεξεργαστή, στοίβας προγράμματος, κτλ);
3. Το πρόγραμμα πρέπει να μειώνει όσο το δυνατό περισσότερο τον χρόνο διεκπεραίωσης της εντολής.
4. Συμβουλευτείτε τις οδηγίες του ΕΠΛ132 για συγγραφή των σχετικών ελέγχων μονάδων, διάσπαση του προγράμματος σε πολλαπλά αρχεία, τεκμηρίωση, makefile, κτλ.

Καλή Επιτυχία !