ΠΑΝΕΠΙΣΤΗΜΙΟ ΚΥΠΡΟΥ
Τμήμα Πληροφορικής
ΕΠΛ 421 - Προγραμματισμός Συστημάτων

ΑΣΚΗΣΗ 3 – Υλοποίηση Κελύφους UCYAISH με Κλήσεις Συστήματος

Διδάσκων: Δημήτρης Ζεϊναλιπούρ
Υπεύθυνος Εργαστηρίου: Παύλος Αντωνίου

Ημερομηνία Ανάθεσης: Πέμπτη 02/11/2023
Ημερομηνία Παράδοσης: Τετάρτη 15/10/2023

(ο κώδικας να υποβληθεί ηλεκτρονικά σε zip 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 artificial intelligent shell – ucyaish) στη γλώσσα C. Αυτό το κέλυφος θα έχει 2 λειτουργίες: (α) θα συμπεριφέρεται παρόμοια με τα κοινά shell, ειδικότερα το bash, με περιορισμένες λειτουργίες (δεν θα υποστηρίζει δομές προγραμματισμού και προχωρημένες λειτουργίες) και (β) θα δίνει τη δυνατότητα στο χρήστη να διατυπώσει ερώτημα σε φυσική γλώσσα και με υποστήριξη υπηρεσιών τεχνητής νοημοσύνης να λαμβάνει το αποτέλεσμα. Οι λειτουργίες του προγράμματος σας και το αναμενόμενο αποτέλεσμα περιγράφονται αναλυτικότερα στην συνέχεια.

II. Περιγραφή των ζητουμένων λειτουργιών του ucyaish

Η πρώτη λειτουργία του κελύφους σας θα είναι η βασική (manual) που θα δέχεται εντολές bash από το χρήστη, θα τις εκτελεί και θα επιστρέφει το αποτέλεσμα. Η δεύτερη λειτουργία με χρήση υπηρεσιών τεχνητής νοημοσύνης (artificial intelligence) θα υποστηρίζεται από το chatGPT. Πιο συγκεκριμένα, θα δέχεται επερωτήματα γραμμένα σε φυσική γλώσσα, θα ζητά από το chatGPT τις εντολές bash που απαντούν στο επερώτημα και στη συνέχεια θα τις εκτελεί και θα δείχνει το αποτέλεσμα στο χρήστη.

III. Βασική λειτουργία (-m)

III-A. 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. Σε σχέση με τις καθολικές μεταβλητές, ο χρήστης του κελύφους σας θα πρέπει να είναι σε θέση να ανακτήσει τις τιμές των καθολικών μεταβλητών όπως για παράδειγμα $PWD, $OLDPWD, $HOSTNAME, $PATH, $HOME, $PS1. Σε σχέση με τις τοπικές μεταβλητές, ο χρήστης του κελύφους σας θα πρέπει να είναι σε θέση να προσθέσει δικές του καθώς και να δει και να αλλάξει οποιαδήποτε από τις υπάρχουσες τοπικές μεταβλητές. Για την ανάκτηση της τιμής καθολικών μεταβλητών καθώς και για την δημιουργία και ανάκτηση της τιμής τοπικών μεταβλητών δείτε την παράγραφο ΙΙ-C πιο κάτω.

Επιπλέον, το κέλυφος σας πρέπει να καλύπτει κάποιες από τις builtin συναρτήσεις του bash που αναφέρονται στις διαφάνειες 8-30 (cd, declare, echo, exit) και 8-31 (history, kill, let, local, logout, pwd, read) χρησιμοποιώντας standard I/O library calls ή system calls. Για παράδειγμα, η εντολή `read -p "Give a number" a` μπορεί να υλοποιηθεί με χρήση της εντολής printf και scanf ενώ η εντολή pwd μπορεί να υλοποιηθεί μέσω της getcwd() που ορίζεται μέσα στη βιβλιοθήκη unistd.h.

ΙΙΙ-Β. Λειτουργίες

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

- Εκτέλεση βασικών εντολών με ορίσματα.
- Δυνατότητα εξόδου από το κέλυφος.
- Να επιτρέπεται η εκτέλεση μιας εντολής 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.
- Για κάθε εντολή, το κέλυφος θα κάνει fork μια νέα διεργασία (process).

Βοηθητικές υποθέσεις:

1. Υποθέστε ότι σε ένα επερώτημα (σύνολο εντολών που δίνεται από το χρήστη) μπορούμε να έχουμε πολλαπλά pipes (|) και (α) αν υπάρχει ανακατεύθυνση σε αρχείο μπορεί να είναι στο τέλος (μετά την τελευταία εντολή) ενώ (β) αν υπάρχει ανακατεύθυνση από αρχείο θα είναι στην αρχή (στην πρώτη εντολή) όπως για παράδειγμα:
   cat /etc/passwd | cut -d":" -f 2 | sort > output.txt
   cat < etc/passwd | cut -d":" -f 2 | sort
2. Υποθέστε ότι η εκτέλεση στο παρασκήνιο θα ζητείται πάνω σε μια εντολή (χωρίς pipes και χωρίς ανακατευθύνσεις) όπως για παράδειγμα:
   find . -type f -name "*.c" &

ΙΙΙ-C. Βοηθητικές Συναρτήσεις και Μεταβλητές

Για την διεκπεραίωση αυτής της βασικής λειτουργίας του shell σας 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

III-D. Παραδείγματα εκτέλεσης εντολής

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

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

```
ucyaish>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.

IV. Λειτουργία τεχνητής νοημοσύνης (-a)

IV-A. Γενική Περιγραφή

Στη δεύτερη λειτουργία, το πρόγραμμα θα δέχεται επερωτήματα σε φυσική γλώσσα όπως για παράδειγμα:

```
print the contents of the file test.c
```

Συνήθως μια επερώτηση προς το chatGPT επιστρέφει πάνω από μία λύσεις για το ζητούμενο εμπλουτισμένες με σχόλια και επεξηγήσεις. Αν το ερώτημα δεν είναι σαφές έτσι ώστε η απάντηση από το ChatGPT να είναι μόνο μια λύση, τότε το shell σας να τυπώνει το μήνυμα "Command cannot be executed! Please specify your query precisely."

Θα πρέπει ο χρήστης να τροποποιήσει το επερώτημα έτσι ώστε να επιστρέφει μια λύση (και όχι πολλές διαφορετικές εκδοχές λύσης) και η απάντηση να περιέχει μόνο εντολές bash και όχι άλλα σχόλια (comments) διότι το chatGPT τείνει να επιστρέφει και επεξηγήσεις για τη/τις λύση/εις που προτείνει. Ένα καλύτερο επερώτημα θα ήταν:

```
Tell me the bash command to print the contents of the file test.c. Give me
only one option. Do not provide any other comments.
```

Για κάθε επερώτηση του χρήστη, το κέλυφός σας θα πρέπει να στέλνει μια αίτηση στο chatGPT μέσω του API έτσι ώστε να λάβει την εντολή ή το σύνολο εντολών του bash το οποίο κάνει αυτό που ζητά ο χρήστης και μετά να το εκτελεί και να παρουσιάζει τα αποτελέσματα στην οθόνη.

Μαζί με την επιλογή `-a` μπορεί να δοθούν και ακόμη μια προαιρετική επιλογές:

`-i` για interactive mode, δηλαδή κάθε φορά που επιστρέφει μια εντολή το ChatGPT, αν το option αυτό είναι ενεργοποιημένο, θα τυπώνει το μήνυμα: "Are you sure you want to execute the command: <command returned by chatGPT> y or n?" και θα αναμένει από το χρήστη να απαντήσει y ή n για να προχωρήσει στην εκτέλεση ή στην απόρριψη της εντολής αντιστοίχως.

Οι επιλογές αυτές αυτά μπορούν να δοθούν με οποιαδήποτε σειρά. Για την επεξεργασία των επιλογών του προγράμματος (command line arguments) σας συνιστούμε ανεπιφύλακτα να κάνετε χρήση της βιβλιοθήκης `getopt`. Για περισσότερες πληροφορίες "`man 3 getopt`", όπου 3 δηλώνει "C library routines". Για παράδειγμα χρήσης μπορείτε να δείτε εδώ. Θα χρειαστεί να ενσωματώσετε τη βιβλιοθήκη getopt.h.

Σε περίπτωση που ο χρήστης επιλεγεί να εκτελέσει μια επιστρεφόμενη εντολή, manually ή interactively, τότε το αποτέλεσμα είναι αυτό που θα εκτελεστεί. Εάν η εντολή είναι λανθασμένη, τότε το λάθος βαραίνει τον χρήστη που επέλεξε να εκτελέσει την εντολή.

IV-B. Προαπαιτούμενα (curl + json string manipulation)

Το πρόγραμμα θα αλληλεπιδρά με το chatGPT με χρήση εντολής curl που μπορεί να στείλει HTTP POST request στο chatGPT API endpoint https://api.openai.com/v1/chat/completions.
Ένα παράδειγμα εντολής curl που μπορείτε να στείλετε από το bash shell για να λάβετε μια απάντηση από το chatGPT είναι η πιο κάτω:

```
curl https://api.openai.com/v1/chat/completions \
  -H "Content-Type: application/json" \
  -H "Authorization: Bearer $OPENAI_API_KEY" \
  -d '{
  "model": "gpt-3.5-turbo",
  "messages": [
    {
      "role": "user",
      "content": ""
    },
    {
      "role": "assistant",
      "content": "... to print the contents of the file test.c ..."
    }
  ],
  "temperature": 1,
  "max_tokens": 256,
  "top_p": 1,
  "frequency_penalty": 0,
  "presence_penalty": 0
}'
```

Για να αποκτήσετε το δικό σας πρέπει να ξεκινήσετε από το πιο κάτω σύνδεσμο https://platform.openai.com/login?launch

Αντικαταστήστε τις ... με ότι κρίνετε χρήσιμο.

Εσείς θα πρέπει να τρέξετε την εντολή curl μέσα από το C πρόγραμμά σας. Θα πρέπει να περιλάβετε την πιο κάτω βιβλιοθήκη `#include <curl/curl.h>`
και κατά για τη μεταγλώττιση με το gcc να δώσετε την επιλογή `-lcurl`
Ο κώδικας που κάνει ακριβώς την ίδια δουλειά με το πιο πάνω είναι:

```
CURL *curl;
CURLcode res;

curl = curl_easy_init();
if(curl) {

curl_easy_setopt(curl,CURLOPT_URL,"https://api.openai.com/v1/chat/completions");

  struct curl_slist *headers = NULL;
  headers = curl_slist_append(headers, "Content-Type: application/json");
  headers = curl_slist_append(headers, "Authorization: Bearer OPENAI_API_KEY");
  curl_easy_setopt(curl, CURLOPT_HTTPHEADER, headers);

  curl_easy_setopt(curl, CURLOPT_POSTFIELDS, "{ \"model\": \"gpt-3.5-turbo\", \"messages\":
[ { \"role\": \"user\", \"content\": \"... print the contents of the file test.c ...\"  }
],  \"temperature\":  1,  \"max_tokens\":  256,  \"top_p\":  1,  \"frequency_penalty\":  0,
\"presence_penalty\": 0 }");
  curl_easy_setopt(curl, CURLOPT_POSTFIELDSIZE, -1L);

  // Perform the request, res will get the return code
  res = curl_easy_perform(curl);

  // Check for errors
  if(res != CURLE_OK)
    fprintf(stderr, "curl_easy_perform() failed: %s\n", curl_easy_strerror(res));
```

```
  // always cleanup
  curl_easy_cleanup(curl);
}
```

Η απόκριση που θα θέλατε να λάβετε από το ChatGPT θα είναι της πιο κάτω μορφής:

```
{
  "id": "chatcmpl-8F6kqM2rHwf9mvJU3P9vkd3P5nEio",
  "object": "chat.completion",
  "created": 1698611576,
  "model": "gpt-3.5-turbo-0613",
  "choices": [
    {
      "index": 0,
      "message": {
        "role": "assistant",
        "content": "cat test.c"
      },
      "finish_reason": "stop"
    }
  ],
  "usage": {
    "prompt_tokens": 35,
    "completion_tokens": 3,
    "total_tokens": 38
  }
}
```

Όπως μπορείτε να δείτε, η απόκριση είναι σε μορφή JSON. Αν θέλετε να εξάγετε με αποδοτικό τρόπο δεδομένα από JSON strings μπορείτε να χρησιμοποιήσετε κάποια third-party βιβλιοθήκη όπως η cJSON Για να χρησιμοποιήσετε τη συγκεκριμένη βιβλιοθήκη, χρειάζεστε (α) τα αρχεία cJSON.c/.h μέσα στον ίδιο κατάλογο που είναι το πρόγραμμά σας, και (β) τη συμπερίληψη της βιβλιοθήκης `#include "cJSON.h"`.

IV-C. Παραδείγματα εκτέλεσης εντολής

Πιο κάτω δίνονται παραδείγματα εκτέλεσης (με κόκκινο η είσοδος του χρήστη).

1ο Παράδειγμα εκτέλεσης

```
b103ws10:/home/faculty/csp5pa1>./ucyaish -a
ucyaish>print the contents of the file test.c
Command cannot be executed! Please specify your query precisely.
ucyaish>Tell me the bash command to print the contents of the file test.c.
Give me only one option. Do not provide any other comments.
#include <stdio.h>
int main() {
   printf("Hello, World!");
   return 0;
}
ucyaish>close the shell
```

2ο Παράδειγμα εκτέλεσης

```
b103ws10:/home/faculty/csp5pa1>./ucyaish -a -i
ucyaish> Tell me the bash command to print the
contents of the file test.c. Give me only one option.
Do not provide any other comments.
Are you sure you want to execute the command: "cat test.c" y or n? y
#include <stdio.h>
int main() {
```

Η εντολή του χρήστη θα μπορούσε να ήταν οποιαδήποτε από τις πιο κάτω που κάνουν ακριβώς την ίδια δουλειά:

```
./ucyaish -i -a
./ucyaish -a -i
./ucyaish -ia
./ucyaish -ai
```

```
   printf("Hello, World!");
   return 0;
}
ucyaish> Tell me the bash command to close the shell. Give me only one
option. Do not provide any other comments.
Are you sure you want to execute the command: "exit" y or n? y
```

Πιο κάτω δίνουμε κάποια περαιτέρω ενδεικτικά prompts χωρίς παρουσιαστικά στοιχεία ucyaish.

Prompt #1

give me a single bash command that creates an sqllite database table named Employee with two attributes, id and name. Please add two records to the database with values (1, Demetris) and (2, Pavlos) and then present all records in the table. Give me only one option. Do not provide any other comments.

```
sqlite3 employees.db "CREATE TABLE Employee (id INT, name TEXT); INSERT
INTO Employee VALUES (1, 'Demetris'), (2, 'Pavlos'); SELECT * FROM
Employee;"

1|Demetris
2|Pavlos
```

Prompt #2

give me a single bash command that creates a text file in which you add the dictionary names starting from a. Give me only one option. Do not provide any other comments.

```
grep -i "^a" /usr/share/dict/words > dictionary_a.txt
```

Prompt #3

give me a single bash command that enumerates uniquely all running processes in shell in ascending sorted order. Give me only one option. Do not provide any other comments.

```
ps -eo user= --no-headers | sort -u
```

V. 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 !