Name : Muhammad Ishraf Shafiq Zainuddin

ID : 200342741

Assgn : 2

Part 1 (Forking)

(a) Based on the CS 330 lab page, a child process is being generated by LINUX by copying the parents process when a fork() system is called. It creates a new process without having to replace the current process. Both of the processes can run a unique task simultaneously. However, it's a different case when exec() system is called since it creates a new process by replacing the current's (cs.uregina.ca/Links/class-info/330/Fork/fork.html).

(b) The program will be slowing down and eventually crash when this particular program being run (fork() insides a while loop). This program is called a fork bomb which is a denial of service (DoS) attack where in a process continually replicates itself to deplete available system resources (wikipedia.org/wiki/Fork_bomb) or when the fork system call is recursively used untill all system is resources execute a command (incapsula.com/ddos/attack-glossary/fork-bomb.html).

Part 2 (The POSIX Specifications)

(a) Based on POSIX spec. standard, the shell which is a command language interpreter has several different operations such as taking input from user or file into tokens and parses them into simple and compound commands. Furthermore, a shell would also be able to execute a function, performing redirection and various expansions
(pubs.opengroup.org/onlinepubs/9699919799//utilities/V3_chap02.html#tag_18_09).

(b) Based on the POSIX spec., the term "built-in" implies that the shell can execute the utility directly and does not need to search for it.
(pubs.opengroup.org/onlinepubs/9699919799//utilities/V3_chap02.html#tag_18_09).

Part 3 (Writing a Shell)

(a) help, madlib, exit

Reference : https://brennan.io/2015/01/16/write-a-shell-in-c/

//Functions declarations

```
int lsh_help(char **args);
int lsh_exit(char **args);


char *builtin_str[] = {
  "help",
  "exit"
};
```

```
int (*builtin_func[]) (char **) = {
  &lsh_help,
  &lsh_exit
};

int lsh_num_builtins() {
  return sizeof(builtin_str) / sizeof(char *);
}
```

//Function Implementation

//Based on the reference, I believe this is the function that takes no arguments but returns the list of built-in commands and information on how the shell works.

```
int lsh_help(char **args)
{
  int i;
  for (i = 0; i < lsh_num_builtins(); i++) {
    printf("  %s\n", builtin_str[i]);
  }

  printf("Help??\n"); //help??
  return 1;
}
```

// Takes no arguments and quits the shell (exit)

```
int lsh_exit(char **args)
{
  return 0;
}
```

//Madlib

Reference : https://rosettacode.org/wiki/Mad_Libs

```
void lsh_madlibs(dstr **args)
{
    static const size_t buffer_size = 128;
    char insert[buffer_size];
    char replace[buffer_size];

    char *start,
         *end = story->data;

    while (start = strchr(end, '<'))
    {
        if (!(end = strchr(start, '>'))) err("Malformed brackets in input");

        strncpy(replace, start, end - start + 1);
        replace[end - start + 1] = '\0';

        printf("Enter value for field %s: ", replace);
```

```
        fgets(insert, buffer_size, stdin);
        const size_t il = strlen(insert) - 1;
        if (insert[il] == '\n')
            insert[il] = '\0';

        dstr_replace_all(story, replace, insert);
    }
    printf("\n");
}
```