Operating System Lab CPE 435

Lab3 – Build Your Own Linux-Shell

Introduction:

The shell or command line interpreter is the fundamental user interface to an operating system, each interactive user can send commands to the OS and by which the OS can respond to the user. The command line is a sequence of ASCII text words delimited by whitespace. The first word in the command line is either the name of a built-in command or the pathname of an executable file. The remaining words are command-line arguments. If the first word is a built-in command, the shell immediately executes the command in the current process. Otherwise, the word is assumed to be the pathname of an executable program. In this case, the shell forks a child process, then loads and runs the program in the context of the child. The child processes created as a result of interpreting a single command line are known collectively.

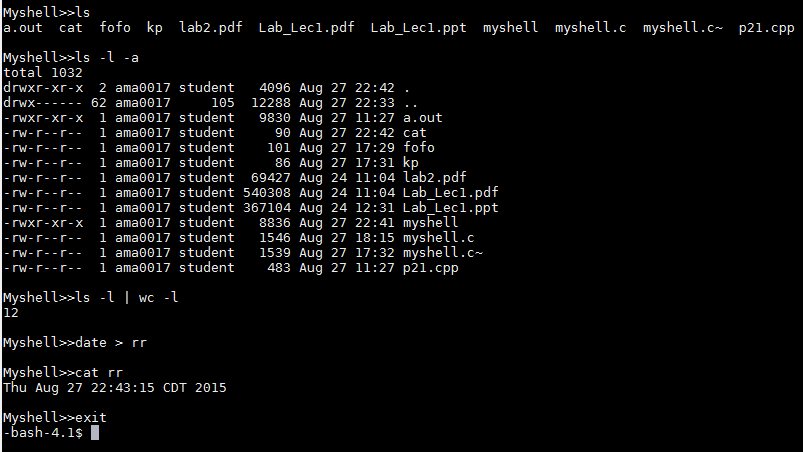
The new shell should supports:

* user commands, such as ls ,date,ls –l –a,
* commands with I/O re-direction ,ex : ls –l > a.txt
* commands with a single pipe ,ex : who | wc –l

Like all Linux shells, your shell executes a loop. It

* prints the shell prompt,
* reads the command line (terminated with NULL)
* Parses the command line and create its arguments.
* execs the command with its arguments, then waits until the command finishes
* Repeat the loop until the exit command is entered.

Sample Output:



* **char \*strtok(char \****str***, const char \****delim***);**

The **strtok**() function parses a string into a sequence of tokens. On the first call to **strtok**() the string to be parsed should be specified in *str*. In each subsequent call that should parse the same string, *str* should be NULL.The *delim* argument specifies a set of bytes that delimit the tokens in the parsed string. The caller may specify different strings in *delim* in successive calls that parse the same string.

* **int dup2(int oldfd, int newfd);**

dup2() makes newfd be the copy of oldfd, closing newfd first if necessary, but note the following:

* If oldfd is not a valid file descriptor, then the call fails, and newfd is not closed.
* If oldfd is a valid file descriptor, and newfd has the same value as oldfd, then dup2() does nothing, and returns newfd.
* **char \*gets(char \*s)**

Reads a line from stdin into the buffer pointed to by s until either a terminating newline or EOF, which it replaces with a null byte (aq\0aq).

* **int pipe(int pipefd[2]);**

pipe () creates a pipe, a unidirectional data channel that can be used for interprocess communication. The array pipefd is used to return two file descriptors referring to the ends of the pipe. pipefd[0] refers to the read end of the pipe. pipefd[1]refers to the write end of the pipe. Data written to the write end of the pipe is buffered by the kernel until it is read from the read end of the pipe.