Resolución de la primera parte

Se editó únicamente los archivos `exec.c` , y `parsing.c` para cumplir con la funcionalidad pedida en la primera parte. Se muestra un diff de la función modificada, `exec_cmd` :

```
                case EXEC:
                        // spawns a command
                        //
-                       // Your code here
-                       printf("Commands are not yet implemented\n");
+                       handle_exec(cmd);
                        _exit(-1);
                        break;


void handle_exec(struct cmd* cmd) {
    struct execcmd* execcmd = (struct execcmd*) cmd;
    execvp(execcmd->argv[0], execcmd->argv);
}
```

Nota: no se valida el código de retorno de `execvp` debido a que se asume que inmediatamente después de la llamada a `handle_exec` habrá un `_exit` , y se manejará el mensaje de error apropiado en el proceso padre.

`parsing.c`

```
static char* expand_environ_var(char* arg) {
        if (arg[0] == '$') {
                char* env = getenv(arg + 1);
                if (!env) {  // Expand into empty variable
                        env = "";
                }
                size_t len = strlen(env) + 1;
                arg = realloc(arg, len);
                memset(arg, 0, len);
                strncpy(arg, env, len);
        }
        return arg;
}
```

Preguntas

- ¿cuáles son las diferencias entre la syscall execve(2) y la familia de wrappers proporcionados por la librería estándar de C (libc) exec(3)?

`execve` es la syscall proporcionada por el sistema operativo, que reemplaza la imagen del proceso actual por la del programa ejecutable, argumentos y variables de entorno pasadas por parámetro. Los wrappers son funciones de la librería estándar de C, y ejecutan internamente `execve` , proporcionando funcionalidades adicionales como búsqueda del ejecutable en el `$PATH` o simplemente la posibilidad de pasar los argumentos de la nueva imagen como parámetros individuales en vez de un único array de punteros a char.