Operating Systems
Jacobs University Bremen
Dr. Jürgen Schönwälder

Course: 320202
Date: 2015-04-10
Deadline: 2015-04-15

Problem Sheet[1] #4

Problem 4.1: *dynamic linking* (10 points)

Many operating systems can dynamically link additional object code into a running process. This allows to design software such that the functionality can be extend by writing plugins that are loaded at runtime. The widely used shell `bash` supports this via the `enable` command.

A simple example demonstrating the usage of the programming interface to the dynamic linking loader can be found in the source code linked to the course web page. The `cat++` program implements a version of `cat` that can apply transformations to the files concatenated. The transformations are dynamically loaded into `cat++`.

Your task is to extend the simple shell `msh` with a builtin command `enable` that mimics the behavior of the same command in the `bash` shell. The following command options must be supported:

- The command `enable` without any arguments prints a list of builtin commands.
- The command `enable -f file [name ...]` loads the dynamically linked object `file` and searches for an implementation of the listed command names in the dynamically linked object file. All commands that were found are added to the list of builtin commands.
- The command `enable -d [name ...]` deletes the listed builtin commands. When the last builtin command of a dynamically linked object file is deleted, the dynamically linked object file is unloaded.

Below is an example execution loading a plugin that provides the builtin commands echo and ohce (a reverse echo).

```
$ ./msh-enable
msh > enable
enable enable
msh > enable -f ./libecho.so echo ohce
msh > enable
enable ohce
enable echo
enable enable
msh > echo 1 2 3
1 2 3
msh > ohce 1 2 3
3 2 1
msh > echo -d echo ohce
msh > enable
enable enable
```

On the next page, you will find the source of the `echo` plugin that implements the commands `echo` and `ohce`.

[1] See the course web page for submission instructions and grading details.

```c
/*
 * msh/echo.c
 *
 *      Implementation of an echo command and a reverse echo
 *      command called oche.
 */

#include <stdio.h>
#include <string.h>

int
cmd_echo(int argc, char **argv)
{
    for (int i = 1; i < argc; i++) {
        printf("%s%s", (i > 1) ? " " : "", argv[i]);
    }
    puts("");

    return 0;
}

int
cmd_ohce(int argc, char **argv)
{
    for (int i = argc-1; i > 0; i--) {
        printf("%s%s", (i < argc-1) ? " " : "", argv[i]);
    }
    puts("");

    return 0;
}
```