Execute a terminal command from a Cocoa app

How can I execute a terminal command (like `grep`) from my Objective-C Cocoa application?

objective-c cocoa osx

edited Feb 3 '11 at 1:41
Jonas
32.2k 69 201 296

asked Jan 5 '09 at 8:21
lostInTransit
31.1k 50 169 248

1 Im just stating the obvious: with sandboxing you can't just go start apps that are not in your sandbox AND they need to be signed by you to allow this – Daij-Djan Aug 27 '15 at 9:05

11 Answers

You can use `NSTask`. Here's an example that would run '`/usr/bin/grep foo bar.txt`'.

```
int pid = [[NSProcessInfo processInfo] processIdentifier];
NSPipe *pipe = [NSPipe pipe];
NSFileHandle *file = pipe.fileHandleForReading;

NSTask *task = [[NSTask alloc] init];
task.launchPath = @"/usr/bin/grep";
task.arguments = @[@"foo", @"bar.txt"];
task.standardOutput = pipe;

[task launch];

NSData *data = [file readDataToEndOfFile];
[file closeFile];

NSString *grepOutput = [[NSString alloc] initWithData: data encoding:
NSUTF8StringEncoding];
NSLog (@"grep returned:\n%@", grepOutput);
```

`NSPipe` and `NSFileHandle` are used to redirect the standard output of the task.

For more detailed information on interacting with the operating system from within your Objective-C application, you can see this document on Apple's Development Center: Interacting with the Operating System.

Edit: Included fix for NSLog problem

If you are using NSTask to run a command-line utility via bash, then you need to include this magic line to keep NSLog working:

```
//The magic line that keeps your log where it belongs
task.standardOutput = pipe;
```

An explanation is here: http://cocoadev.com/HowToPipeCommandsWithNSTask

edited Jun 9 '15 at 7:14
Stunner
5,453 6 42 102

answered Jan 5 '09 at 8:28
Gordon Wilson
19.5k 8 49 54

1 Yup, 'arguments = [NSArray arrayWithObjects: @"-e", @"foo", @"bar.txt", nil];' – Gordon Wilson Jan 5 '09 at 8:47

12 There's a small glitch in your answer. NSPipe has a buffer (set at the OS level), which is flushed when it's read. If the buffer fills up, NSTask will hang, and your app will hang too, indefinitely. No error message will appear. This can happen if the NSTask returns a lot of info. The solution is to use `NSMutableData *data = [NSMutableData dataWithCapacity:512];` . Then, `while ([task isRunning]) { [data appendData:[file readDataToEndOfFile]]; }` . And I "believe" you should have one more `[data appendData:[file readDataToEndOfFile]];` after the while-loop exits. – Dave Gallagher Sep 27 '11 at 22:49

1 Errors won't come up unless you do this (they just get printed in the log): [task setStandardError:pipe]; – Mike Sprague Aug 7 '12 at 22:32

1 This could be updated with ARC and with Obj-C array literals. E.g. pastebin.com/sRvs3CqD – bames53 Sep 25 '13 at 23:39

1 It's also a good idea to pipe the errors. `task.standardError = pipe;` – itsdavyh Oct 20 '14 at 23:44

in the spirit of sharing... this is a method I use frequently to run shell scripts. you can add a script to your product bundle (in the copy phase of the build) and then have the script be read and run at runtime. note: this code looks for the script in the privateFrameworks sub-path. warning: this could be a security risk for deployed products, but for our in-house development it is an easy way to customize simple things (like which host to rsync to...) without re-compiling the application, but just editing the shell script in the bundle.

```
//------------------------------------------------------
-(void) runScript:(NSString*)scriptName
{
    NSTask *task;
    task = [[NSTask alloc] init];
    [task setLaunchPath: @"/bin/sh"];

    NSArray *arguments;
    NSString* newpath = [NSString stringWithFormat:@"%@/%@",[[NSBundle mainBundle]
privateFrameworksPath], scriptName];
    NSLog(@"shell script path: %@",newpath);
    arguments = [NSArray arrayWithObjects:newpath, nil];
    [task setArguments: arguments];

    NSPipe *pipe;
    pipe = [NSPipe pipe];
    [task setStandardOutput: pipe];

    NSFileHandle *file;
    file = [pipe fileHandleForReading];

    [task launch];

    NSData *data;
    data = [file readDataToEndOfFile];

    NSString *string;
    string = [[NSString alloc] initWithData: data encoding: NSUTF8StringEncoding];
    NSLog (@"script returned:\n%@", string);
}
//------------------------------------------------------
```

Edit: Included fix for NSLog problem

If you are using NSTask to run a command-line utility via bash, then you need to include this magic line to keep NSLog working:

```
//The magic line that keeps your log where it belongs
[task setStandardInput:[NSPipe pipe]];
```

In context:

```
NSPipe *pipe;
pipe = [NSPipe pipe];
[task setStandardOutput: pipe];
```

```
//The magic line that keeps your log where it belongs
[task setStandardInput:[NSPipe pipe]];
```

An explanation is here: http://www.cocoadev.com/index.pl?NSTask

edited Mar 20 '10 at 11:19
Steve McLeod
25.5k 29 92 139

answered Mar 30 '09 at 12:10
kent
2,758 1 17 29

1 The explanation link is dead. – Jonny Jul 8 '14 at 4:06

I want to run this command "system_profiler SPApplicationsDataType -xml" but i am getting this error "launch path not accessible" – Vikas Bansal Jul 29 '15 at 12:20

---

kent's article gave me a new idea. this runCommand method doesn't need a script file, just runs a command by a line:

```
- (NSString *)runCommand:(NSString *)commandToRun
{
    NSTask *task = [[NSTask alloc] init];
    [task setLaunchPath:@"/bin/sh"];

    NSArray *arguments = [NSArray arrayWithObjects:
                          @"-c" ,
                          [NSString stringWithFormat:@"%@", commandToRun],
                          nil];
    NSLog(@"run command:%@", commandToRun);
    [task setArguments:arguments];

    NSPipe *pipe = [NSPipe pipe];
    [task setStandardOutput:pipe];

    NSFileHandle *file = [pipe fileHandleForReading];

    [task launch];

    NSData *data = [file readDataToEndOfFile];

    NSString *output = [[NSString alloc] initWithData:data
encoding:NSUTF8StringEncoding];
    return output;
}
```

You can use this method like this:

```
NSString *output = runCommand(@"ps -A | grep mysql");
```

edited May 17 at 0:31

answered Sep 7 '12 at 0:15
Kenial
1,188 14 18

1 works like a charm thanks.. – kid Jul 14 '15 at 10:51

This handles most cases well, but if you run it in a loop, it eventually raises an exception due to too many open file handles. Can be fixed by adding: [file closeFile]; after readDataToEndOfFile. – David Stein May 15 at 19:54

@DavidStein : I think using autoreleasepool to wrap runCommand method seems to be rather than. Actually, above code doesn't consider non-ARC as well. – Kenial May 17 at 0:39

@Kenial: Oh, that's a much better solution. It also releases the resources promptly upon leaving the scope. – David Stein May 19 at 7:02

---

fork, exec, and wait should work, if you're not really looking for a Objective-C specific way. `fork` creates a copy of the currently running program, `exec` replaces the currently running program with a new one, and `wait` waits for the subprocess to exit. For example (without any error checking):

```
pid_t p = fork();
if (p == 0) {
    /* fork returns 0 in the child process. */
    execl("/other/program/to/run", "/other/program/to/run", "foo", NULL);
} else {
    /* fork returns the child's PID in the parent. */
    int status;
    wait(&status);
    /* The child has exited, and status contains the way it exited. */
}

/* The child has run and exited by the time execution gets to here. */
```

There's also system, which runs the command as if you typed it from the shell's command line. It's simpler, but you have less control over the situation.

I'm assuming you're working on a Mac application, so the links are to Apple's documentation for these functions, but they're all `POSIX`, so you should be to use them on any POSIX-compliant system.

answered Jan 5 '09 at 8:47
Zach Hirsch
7,230 5 19 27

I know this is an very old answer but i need to say this: this is an excelent way to use trheads to handle the excecution. the only downside is that it creates a copy of the entire program. so for a cocoa application i would go with @GordonWilson for a nicer aproach, and if i'm working on a command line application this is the best way to do it. thanks (sorry my bad english) – Nicos Karalis Feb 17 '13 at 21:26

There is also good old POSIX system("echo -en '\007'");

edited May 25 '13 at 13:15

answered Jan 13 '09 at 12:54
nes1983
8,363 2 32 50

4 DO NOT RUN THIS COMMAND. (In case you do not know what this command does) – justin Dec 8 '09 at 23:18

4 Changed it to something slightly safer … (it beeps) – nes1983 Dec 8 '09 at 23:23

Won't this throw an error in the console? `Incorrect NSStringEncoding value 0x0000 detected. Assuming NSStringEncodingASCII. Will stop this compatibility mapping behavior in the near future.` – cwd Dec 17 '11 at 18:47

1 Hmm. Maybe you have to double-escape the backslash. – nes1983 Dec 17 '11 at 19:37

Cleaned up the code in the top answer to make it more readable, less redundant, added the benefits of the one-line method and made into an NSString category

```
@interface NSString (ShellExecution)
- (NSString*)runAsCommand;
@end
```

Implementation:

```
@implementation NSString (ShellExecution)

- (NSString*)runAsCommand {
    NSPipe* pipe = [NSPipe pipe];

    NSTask* task = [[NSTask alloc] init];
    [task setLaunchPath: @"/bin/sh"];
    [task setArguments:@[@"-c", [NSString stringWithFormat:@"%@", self]]];
    [task setStandardOutput:pipe];

    NSFileHandle* file = [pipe fileHandleForReading];
    [task launch];

    return [[NSString alloc] initWithData:[file readDataToEndOfFile]
encoding:NSUTF8StringEncoding];
}

@end
```

Usage:

```
NSString* output = [@"echo hello" runAsCommand];
```

And **if** you're having problems with output encoding:

```
// Had problems with `lsof` output and Japanese-named files, this fixed it
NSString* output = [@"export LANG=en_US.UTF-8;echo hello" runAsCommand];
```

Hope it's as useful to you as it will be to future me. (Hi, you!)

edited Sep 25 '13 at 23:16

answered Sep 25 '13 at 20:20
inket
963 10 15

1 Indeed, your code was very useful to me! I changed it to Swift and posted it as another answer below. – ElmerCat Aug 27 '15 at 3:14

I wrote this "C" function, because `NSTask` is obnoxious..

```
NSString * runCommand(NSString* c) {

    NSString* outP; FILE *read_fp;  char buffer[BUFSIZ + 1];
    int chars_read; memset(buffer, '\0', sizeof(buffer));
    read_fp = popen(c.UTF8String, "r");
    if (read_fp != NULL) {
        chars_read = fread(buffer, sizeof(char), BUFSIZ, read_fp);
        if (chars_read > 0) outP = $UTF8(buffer);
        pclose(read_fp);
    }
    return outP;
}

NSLog(@"%@", runCommand(@"ls -la /"));

total 16751
drwxrwxr-x+ 60 root        wheel     2108 May 24 15:19 .
drwxrwxr-x+ 60 root        wheel     2108 May 24 15:19 ..
…
```

oh, and for the sake of being complete / unambiguous…

```
#define $UTF8(A) ((NSString*)[NSS stringWithUTF8String:A])
```

Years later, `C` is still a bewildering mess, to me.. and with little faith in my ability to correct my gross shortcomings above - the only olive branch I offer is a rezhuzhed version of @inket's answer that is *barest of bones*, for my fellow purists / verbosity-haters...

```
id _system(id cmd) {
   return !cmd ? nil : ({ NSPipe* pipe; NSTask * task;
  [task = NSTask.new setValuesForKeysWithDictionary:
    @{ @"launchPath" : @"/bin/sh",
        @"arguments" : @[@"-c", cmd],
   @"standardOutput" : pipe = NSPipe.pipe}]; [task launch];
  [NSString.alloc initWithData:
     pipe.fileHandleForReading.readDataToEndOfFile
                      encoding:NSUTF8StringEncoding]; });
}
```

edited Apr 1 '15 at 8:54

answered May 24 '13 at 21:00
alex gray
8,483 4 56 87

1 outP is undefined on any error, chars_read is too small for the return value of fread() on any architecture where sizeof(ssize_t) != sizeof(int), what if we want more output than BUFSIZ bytes? What if the output isn't UTF-8? What if pclose() returns an error? How do we report the error of fread()? – ObjectiveC-oder Mar 11 '14 at 10:45

@ObjectiveC-oder D'oh - I dunno. Please, tell me (as in.. edit away)! – alex gray Dec 2 '14 at 1:30

Here's how to do it in Swift

This is based on inkit's answer above. He wrote it as a category on `NSString`. In Swift, it's written as an extension of `String`.

```
extension String {
    func runAsCommand() -> String {
        let pipe = NSPipe()
        let task = NSTask()
        task.launchPath = "/bin/sh"
        task.arguments = ["-c", String(format:"%@", self)]
        task.standardOutput = pipe
        let file = pipe.fileHandleForReading
        task.launch()
        if let result = NSString(data: file.readDataToEndOfFile(), encoding:
NSUTF8StringEncoding) {
            return result as String
        }
        else {return "--- Unable to initialize string from file data ---"}
    }
}
```

Usage:

```
let output = "echo hello".runAsCommand()
```

or

```
let output = "export LANG=en_US.UTF-8;echo hello".runAsCommand()
```

Note that I first tried initializing a Swift String directly from the data, but that didn't work as expected. Instead of `hello` it created a hexadecimal representation of the string:

```
(<68656c6c 6f0a>, 4)
```

Initializing it as an `NSString` solved that problem, however the `NSString` initialization returns an optional value - it's possible for it to fail and return nil if there's something wrong with the encoding - though, that's unlikely to happen for this case.

The function could have been written to return an optional `String?`, but that would be awkward to use and wouldn't serve a useful purpose. So, to cover that remote possibility, it returns an error message. Otherwise, the `NSString` `result` is returned, cast `as` a Swift `String`.

answered Aug 27 '15 at 3:12
ElmerCat
1,091 1 7 20

If the Terminal command requires Administrator Privilege (aka sudo), use AuthorizationExecuteWithPrivileges instead. The following will create a file named "com.stackoverflow.test" is the root directory "/System/Library/Caches".

```
AuthorizationRef authorizationRef;
FILE *pipe = NULL;
OSStatus err = AuthorizationCreate(nil,
                                   kAuthorizationEmptyEnvironment,
                                   kAuthorizationFlagDefaults,
                                   &authorizationRef);

char *command= "/usr/bin/touch";
char *args[] = {"/System/Library/Caches/com.stackoverflow.test", nil};

err = AuthorizationExecuteWithPrivileges(authorizationRef,
                                         command,
                                         kAuthorizationFlagDefaults,
                                         args,
                                         &pipe);
```

answered Jul 11 '11 at 0:46
SwiftArchitect
13.8k 4 44 78

3 This has been officially deprecated since OS X 10.7 – Sam Washburn Dec 16 '12 at 9:58

Or since Objective C is just C with some OO layer on top you can use the posix conterparts:

```
int execl(const char *path, const char *arg0, ..., const char *argn, (char *)0);
int execle(const char *path, const char *arg0, ..., const char *argn, (char *)0,
char *const envp[]);
int execlp(const char *file, const char *arg0, ..., const char *argn, (char *)0);
int execlpe(const char *file, const char *arg0, ..., const char *argn, (char *)0,
char *const envp[]);
int execv(const char *path, char *const argv[]);
int execve(const char *path, char *const argv[], char *const envp[]);
int execvp(const char *file, char *const argv[]);
int execvpe(const char *file, char *const argv[], char *const envp[]);
```

They are included from unistd.h header file.

answered Jan 5 '09 at 8:32
Paulo Lopes
1,959 14 20

Custos Mortem said:

> I'm surprised no one really got into blocking/non-blocking call issues

For blocking/non-blocking call issues regarding `NSTask` read below:

> asynctask.m -- sample code that shows how to implement asynchronous stdin, stdout & stderr streams for processing data with NSTask

Source code of asynctask.m is available at GitHub.

edited Jun 6 '14 at 8:25
marshaul
150 1 12

answered Jan 28 '11 at 11:37
jon
21 1

protected by Daij-Djan Aug 27 '15 at 9:06

Thank you for your interest in this question. Because it has attracted low-quality or spam answers that had to be removed, posting an answer now requires 10 reputation on this site (the association bonus does not count).

Would you like to answer one of these unanswered questions instead?