



Open Science Grid

Software for DHTC Part 2: Interpreted Languages

Christina Koch (ckoch5@wisc.edu)

Research Computing Facilitator

University of Wisconsin - Madison

Recap

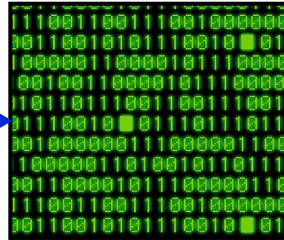
- Previous techniques:
 - Compiled code
 - Download compiled binaries
 - Build yourself
 - Wrapper script
 - Run compiled code
 - Use a pre-built installation ← we're going to explore this further in this session

Interpreted code

- Instead of being compiled and then run...

```
function IsLoggedIn() {
    $pass = $_POST['password'];
    $file = "login.dat";
    $line = fopen($file, "r");
    while ($line = fgets($line)) {
        // User gevonden, password is nu
        $pass = fgets($line);
        break; // stop met de 'for'
    }
    return $pass;
}

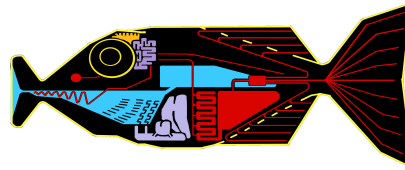
function IsLoggedIn() {
    global $username, $password;
    $login = fopen($file, "r");
    while ($line = fgets($login)) {
        $username = fgets($line);
        $password = fgets($line);
        if ($username == $username && $password == $password) {
            return TRUE;
        }
    }
    return FALSE;
}
```



- ...interpreted languages are translated into binary code “on the fly”

```
function IsLoggedIn() {
    $pass = $_POST['password'];
    $file = "login.dat";
    $line = fopen($file, "r");
    while ($line = fgets($line)) {
        // User gevonden, password is nu
        $pass = fgets($line);
        break; // stop met de 'for'
    }
    return $pass;
}

function IsLoggedIn() {
    global $username, $password;
    $login = fopen($file, "r");
    while ($line = fgets($login)) {
        $username = fgets($line);
        $password = fgets($line);
        if ($username == $username && $password == $password) {
            return TRUE;
        }
    }
    return FALSE;
}
```

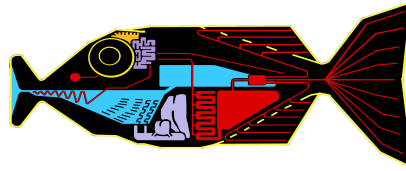


Interpretation

Script

```
getPasswd() {  
  local I=0  
  local line  
  local user  
  local password  
  local file="login.dat"  
  local users=$(cat $file | tr -d '\n')  
  local users_count=$(echo $users | wc -w)  
  while [ $I -lt $users_count ]; do  
    user=$(echo $users | cut -d ' ' -f $I)  
    password=$(cat $file | tr -d '\n' | sed -n "$((I * 2 + 1))p")  
    if [ $(echo $user | grep -q "$password") ]; then  
      echo "User $user found. Password is $password"  
      return $password  
    fi  
    I=$((I + 1))  
  done  
  return FALSE  
}
```

Interpreter

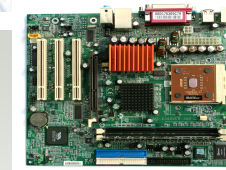


text turns
into binary
instructions

uses



Libraries



On the command line



```
ckoch — bash — 53x14
bash
ckoch5@submit-5:~ ...
ckoch5@os...ster/osg-ss ...
ckoch5@os.../osg/python ... +

[~]$ cat hello.py
import sys

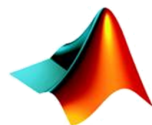
name = sys.argv[1]
print "Hello", name
[~]$ python hello.py "Open Science Grid"
Hello Open Science Grid
[~]$
```

Common interpreted languages*

- Python
- R
- Julia
- Ruby
- Matlab
- Perl
- Javascript



Perl



MATLAB



python™



*Note: the line between interpreted/compiled languages can be fuzzy. Many languages support both options, with one method being more common.

Running interpreted code in jobs

General procedure

- Need to bring along interpreter and script
- Use a wrapper script as the executable
- Wrapper script will:
 - “Install” the interpreter
 - Run the script using the local installation

Python on DHTC

1. Create a portable Python installation (optional)
2. Bring along:
 - pre-built installation OR Python source code
 - your Python code
3. Use a wrapper script to:
 - unpack pre-built install OR install from source
 - run your Python script

Exercises

- Running Python Jobs
 - Exercise 4.1: Pre-building Python and using that installation
 - Exercise 4.2 (optional): Further Python job customizations
- Work on other exercises from today/ yesterday that you weren't able to finish

Questions?

- Now: Hands-on Exercises
 - 3:45 - 5:00pm