Copy of PRACTICING PIPES

Redirecting output

Challenge description asks to use output redirection to write `PWN` to file `COLLEGE`

This can be done by:
`echo PWN > COLLEGE`

FLAG-`pwn.college{8uTmg58URbo6ClnFRLrXgeIbzpW.dRjN1QDL3YjN0czW}`

Redirecting more output

Challenge asks to use output redirectiont to write the output of `/challenge/run` in file `myflag`

This can be done by, `/challenge/run > myflag`

The flag is read by `cat myflag`, which gives the `flagpwn.college{0zmklYETCPAxDCnIO62VlOxFFyg.dVjN1QDL3YjN0czW}`

Appending output

output of `/challenge/run` must be directed to `~/the-flag` in append mode. If it is not in append mode second half of flag will overwrite the first half since first half is automatically written to the file and second half is directed to the file.

The required command is `/challenge/run >> ~/the-flag`

`cat the-flag`

flag-`pwn.college{UK7LV4WXxnhsDz_EtT3aUIkMtaq.ddDM5QDL3YjN0czW}`

Redirecting errors

Output redirected to `myflag` like one of the previous challenges. Errors must be directed to `instructions`. This can be done using

```
/challenge/run > myflag 2> instructions

```

`cat myflag` : `pwn.college{EqCm2ZqoC11xfjUlLqquhsp-xp3.ddjN1QDL3YjN0czW}`

Redirecting input

First, output of `echo COLLEGE` is redirected to the file `PWN` using `echo COLLEGE > PWN`

Then, the file `PWN` is redirected as input to `/challenge/run` using `/challenge/run < PWN`

FLAG-`pwn.college{MXzp8K6r4VK2zTvtiJukYICn0kL.dBzN1QDL3YjN0czW}`

Grepping stored results

Output of `/challenge/run` is redirected to `/tmp/data.txt` using `/challenge/run > /tmp/data.txt`

The flag is searched using grep by the command `grep pwn.college /tmp/data.txt` since all flags start with `pwn.college`

FLAG-`pwn.college{82CDQbBYaKCGAwaGf8297kSq_7G.dhTM4QDL3YjN0czW}`

`[INFO] WELCOME! This challenge makes the following asks of you:`
`[INFO] - the challenge checks for a specific process at the other end of stdout : grep`

`[INFO] - the challenge will output a reward file if all the tests pass : /challenge/.data.txt`

`[HYPE] ONWARDS TO GREATNESS!`

`[INFO] This challenge will perform a bunch of checks.`
`[INFO] If you pass these checks, you will receive the /challenge/.data.txt file.`

`[TEST] You should have redirected my stdout to another process. Checking...`
`[TEST] Performing checks on that process!`

`[INFO] The process' executable is /nix/store/xpq4yhadyhazkcsggmqd7rsgvxb3kjy4-gnugrep-3.11/bin/grep.`
`[INFO] This might be different than expected because of symbolic links (for example, from /usr/bin/python to /usr/bin/python3 to /usr/bin/python3.8).`
`[INFO] To pass the checks, the executable must be grep.`

`[PASS] You have passed the checks on the process on the other end of my stdout!`
`[PASS] Success! You have satisfied all execution requirements.`
`pwn.college{0W_C27US-KC29yxzcZQ-9Wp4CJE.dlTM4QDL3YjN0czW}`

Grepping errors

Similar to previous challenge but errors must be grepped instead.

Standard error redirected to standard output first using `2>& 1`

Therefore the command is,

```
/challenge/run 2>& 1 | grep pwn.college

```

The flag outputted is: `pwn.college{kpH5tI70t8DC93FrTDo270h1fPW.dVDM5QDL3YjN0czW}`

Duplicating piped data with tee

First, `tee` is used to intercept between `/challenge/pwn` and `/challenge/college` so we can find out what is going on. This is done by the command `/challenge/pwn | tee check | /challenge/college`

`Usage: /challenge/pwn --secret [SECRET_ARG]`

`SECRET_ARG should be "I7KzRAn1"`

Therefore, the correct command to attain the flag is `/challenge/pwn --secret IVCKUKeN | /challenge/college` which outputs the following,

`/challenge/pwn --secret I7KzRAn1 | /challenge/college`
`bash: I7KzRAn1: command not found`
`Processing...`
`Correct! Passing secret value to /challenge/college...`
`Great job! Here is your flag:`
`pwn.college{I7KzRAn1yt8uSjLcJcJa6dvoxuj.dFjM5QDL3YjN0czW}`

Writing to multiple programs

Output from `/challenge/hack` must be piped to the 2 commands `/challenge/the` and `/challenge/planet`.

Since `>(/challenge/the)` will give a temporary file which the command `/challenge/the` will read from, it can be used to make `tee` pipe to 2 commands.

The required command is,

```
/challenge/hack | tee >(/challenge/the) | /challenge/planet
```

flag- `pwn.college{UTcjjnAZQrmumViVRMx44WTfCj_.dBDO0UDL3YjN0czW}`

Split-piping stderr and stdout

This was a bit challenging to think of but I eventually found the right answer.

I first used `2>` along with `>(/challenge/the)` to pipe stderr to the `/challenge/the` program

Then I used the standard `|` that has been used in the last couple challenges to pipe stdout to `/challenge/planet`

```
/challenge/hack 2> >(/challenge/the) | /challenge/planet

```

This gives the output,

`/challenge/hack 2> >(/challenge/the) | /challenge/planet`
`Congratulations, you have learned a redirection technique that even experts`
`struggle with! Here is your flag:`
`pwn.college{8SfsR5ePrv1-CKjPWphjSHWn5G1.dFDNwYDL3YjN0czW}`