Programming Tips

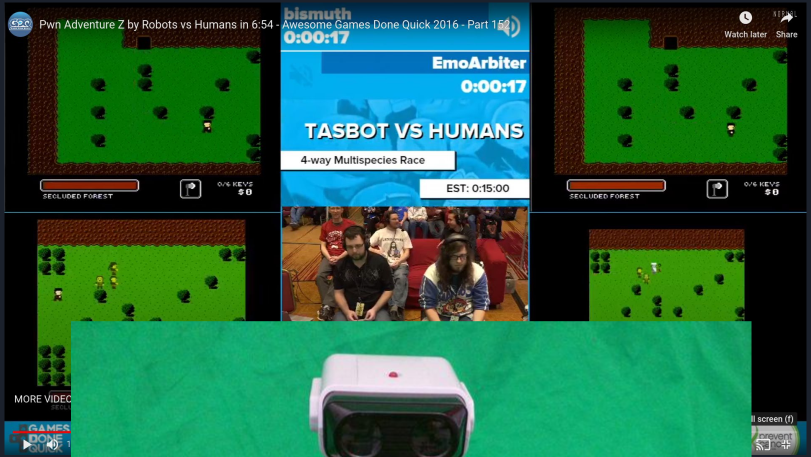
Level 0x04

Quick Overview

- Fun Stuff
- stderr
- Python input

Cybersecurity and Video Game Speedrunning

- West Shore Career Fair Expo
- Presenters
  - Jordan Wiens
  - Michael Wales
- Discussing the overlap of video game speedrunning and cyber security research
- Talk about our work in cyber security



Jeri Ellsworth - Maker / Hacker

- Commodore 64 Kid
- HAM radio (built her own walkie talkies, blew way past FCC limits…)
- Race car driver / chassis builder / rule ~~breaker~~ extender
- Entrepreneur (made her own computer store / chain)
- Made her own transistors
- Made her own 1-chip custom chip C64 in a controller in 1 year
- Worked at Valve (VR / AR / everything research)
- Commodore 64 bass guitar
- Tilt Five - AR Board Game

Lady Ada - Limor Fried

- Bachelors and Masters in Electrical Engineering and Computer Science from MIT
- Founded Adafruit Industries in 2005
- Champion of
  - Maker movement
  - Open Source HW
  - Wearable electronics
- Adafruit sells the best electronic kits
- Best tutorials on how to use new kits / hardware
- Libraries for arduino / Raspberry Pi

Maker Faire

Maker Faire® Orlando

- November 4th and 5th
- Central Florida Fair Grounds
- $15
- Exhibits
  - 3D printers galore
  - Robot wars / battle bots
  - A lot of art exhibits / usually a little more techie than standard art fair
  - Tesla coils

stdin vs stderr

- Different output streams you can write to from your program, both show up in shell
- stderr traditionally used for error messages
  - Typically unbuffered
- stdout used for normal / nominal output

C / C++	Python3

printf(“data for stdout\n”);
std::cout << “data for stdout” << std::endl
	print(“data for stdout”)
	
	import sys
	sys.stdout.write(“data for stdout\n”)

fprintf(stderr, “data for stderr\n”);
std::cerr << “data for stderr” << std::endl;
	import sys
	sys.stderr.write(“data for stderr\n”)

Typical Uses

- If your program has lots of output, errors or warnings might be easily missed

-
  ```
  ./my_program 2>error_log.txt
  Normal output here
  ```

-
  ```
  cat error_log.txt
  Configuration error, using default security settings
  ```
- Compiler errors warnings use stderr
  - Easy to lose warnings when compiling a lot of files
  - Can miss errors when you are compiling many files in parallel
  - Wait till you see the pages that GCC spews when your C++ code goes wrong…

Configurable Debug Output

C / C++

```c
#include <stdio.h>

#ifdef DEBUG
        #define debug(...) fprintf(stderr, __VA_ARGS__)
#else
        #define debug(...) if(0) fprintf(stderr, __VA_ARGS__)
#endif

// Compile with debug enabled
// gcc -D DEBUG test.c
```

Python

```python
import sys

def debug(msg):
  if (True):
    sys.stderr.write(msg + “\n”)
```

```
$ cat test.c
#include <stdio.h>

#ifdef DEBUG
        #define debug(...) fprintf(stderr, __VA_ARGS__)
#else
        #define debug(...) if(0) fprintf(stderr, __VA_ARGS__)
#endif

int main(int argc, char** argv)
{

        printf("Hello World\n");
        debug("This is an error\n");
        return 0;
}


$ gcc -D DEBUG test.c
$ strings a.out | grep This
This is an error
$ gcc test.c
$ strings a.out | grep This
```

Reading input with Python

- Typically don’t want the prompt part of input

```
singleLine = input(“give me data:”)
```

- Instead, typically use

```
singleLine = sys.stdin.readline()
allText = sys.stdin.read()
# both typically have \n on the end, can remove with .strip()
singleLine = sys.stdin.readline().strip()
```

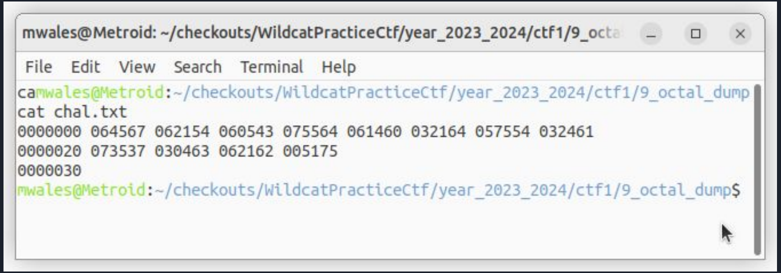
- To read in an integer:

```
myvar = int(sys.stdin.readline())
```

Python Split

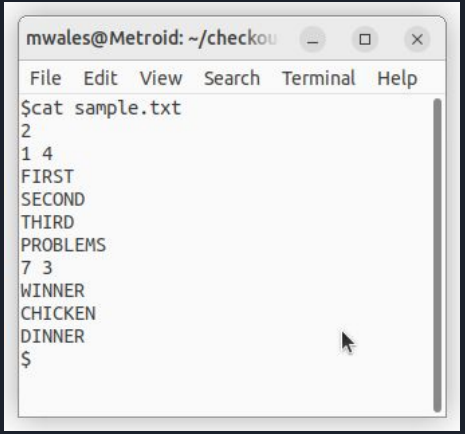
```
>>> print(bunchOfText)
This is a bunch
of text on multiple
lines
>>> bunchOfText.split("\n")
['This is a bunch', 'of text on multiple', 'lines']
>>> bunchOfText.split()
['This', 'is', 'a', 'bunch', 'of', 'text', 'on', 'multiple', 'lines']
```

```
#!/usr/bin/env python3

import sys

allLines = sys.stdin.read().strip()

octalData = []
for singleLine in allLines.split("\n"):
        singleLineParts = singleLine.split(" ")
        singleLineParts.pop(0)

        for eachItem in singleLineParts:
                octalData.append(eachItem)

print("OD: {}".format(octalData))


# When I run this
# cat chal.txt | ./solve.py
# OD: ['064567', '062154', '060543', '075564', '061460', '032164',
'057554', '032461', '073537', '030463', '062162', '005175']
```


mwales@Metroid: ~/checkouts/WildcatPracticeCtf/year_2023_2024/ctf1/9_octa
File Edit View Search Terminal Help

```
camwales@Metroid:~/checkouts/WildcatPracticeCtf/year_2023_2024/ctf1/9_octal_dump
cat chal.txt
0000000 064567 062154 060543 075564 061460 032164 057554 032461
0000020 073537 030463 062162 005175
0000030
mwales@Metroid:~/checkouts/WildcatPracticeCtf/year_2023_2024/ctf1/9_octal_dump$
```

```python
#!/usr/bin/env python3

import sys

def processTestCase():
        print("Do TC")
        firstLine = sys.stdin.readline()

        firstLineParts = firstLine.split(" ")
        firstNum = int(firstLineParts[0])
        secondNum = int(firstLineParts[1])

        print("1st = {} and 2nd = {}".format(firstNum, secondNum))

        wordList = []
        for i in range(secondNum):
                wordList.append(sys.stdin.readline().strip())

        print(wordList)



numTestCases = int(sys.stdin.readline())

print("Number of test cases = {}".format(numTestCases))

for i in range(numTestCases):
        processTestCase()
```



```
$cat sample.txt
2
1 4
FIRST
SECOND
THIRD
PROBLEMS
7 3
WINNER
CHICKEN
DINNER
$
```

Attributions

- https://www.makerfaireorlando.com/
-