CSC352 Spring 2017
Introduction to Interrupts

Weeks 1&2

Dominique Thiébaut
dthiebaut@smith.edu
References

• See class page for references:

• http://www.science.smith.edu/dftwiki/index.php/CSC352_Class_Page_2017
Simplified view of Computation
Simplified view of Computation

Photo credits: https://technick.net/img/guide_uml/guide_uml_016.jpg
Example Program 1

This is the text for an editor that gets characters from the keyboard, and saves and closes the program on Ctrl-X.

... 
init();
...

while (true) {
    while (!has_char()) ;
    ch = get_char();
    if (ch == ^X) {
        savefile();
        exit (0);
    } else
        ...
    } else
        ...
} else
    if ('z' >= ch && 'a' <= ch) {
        insertChar(ch);
    }
}
Example Program 2

This is the same text for an editor, but more contemporary. What's different?

... 
CreateWindow();
EnableEvent(WM_CLOSE);
...

void eventOccurred(Event e) {
    switch (e.code) {
    case WM_CLOSE:
        savefile();
        exit(0);
    case 'a' - 'z':
        insertChar(e.code);
        break;
    default:
        break;
    }
}
The Reality
The Reality

Time

Time

Time
How do Interrupts Work?

- Hardware
- Processor
- Stack
Infrastructure

Processor

I/O Controller

Ram
Infrastructure

Processor

I/O Controllers

Ram
Response to an Interrupt

• At every new instruction:
  
  • if interrupt pending and interrupts allowed…
How fast?

- How fast is a context switch, approximately?
What's a more accurate graph?
We stopped here last time...
• The operating system typically allows programs/processes to run for a fixed amount of time before another process takes over the processor. How can this be implemented?
That's the root of Parallelism!

Image credits:
Process vs Threads
Goals of Multithreading

• Enhance performance

• Increase throughput

• Divide the work into well defined tasks that can be idle waiting for information

• Greater user responsiveness
Memory View

Process

RAM

RAM

Process with Threads
Exploring MultiThreading in Python

https://www.pinterest.com/explore/python-programming/
Caveat

• Python supports multithreading, and multiprocessing.

• Python threads CANNOT RUN IN PARALLEL (GIL)

• If parallelism is needed in Python, use the Multiprocessing library

• Discussion: http://stackoverflow.com/questions/3044580/multiprocessing-vs-threading-python
Examples

• Go to class Web page:
  http://www.science.smith.edu/dftwiki/index.php/Python_Multithreading/Multiprocessing_Examples
Monte-Carlo Pi

\[
m = 281 \\
n = 233 \\
\Pi = 4 \times \frac{n}{m} \\
\Pi = 3.316725978647687
\]

http://montepie.herokuapp.com/
from __future__ import print_function
from random import random

N = 1000000  # int( input( "> " ) )
inside = 0
for i in range( N ):
    x = random()
    y = random()
    if x*x + y*y < 1:
        inside += 1

    if i > 0 and i%1000 == 0:
        print( "%9d %1.12f" % ( i, 4.0*inside/i ) )
Lab, Part 1

Write a multiprocessing application in Python that computes an approximation of Pi using the Monte Carlo simulation, and using 10 Processes.

*Note: You'll have to run your program from the command line!*
Lab, Part 2

• Make your program take as input (command line) the number of multi-processes.

• Measure the execution time of the serial version.

• Measure the execution times of the multiprocessing version with 1, 2, 4, 8, and 16 processes

• Class discussion on the results.
MultiThreading in Java

https://javantura.com/java-logo-background-png/
Java Code
(serial & parallel)

• Go to Class Page: http://www.science.smith.edu/dftwiki/index.php/CSC352:_Computing_Pi_in_Parallel_with_Java