COMP1005 Week 1 Cheat Sheet

Lisa Luff

8/12/2020

Contents

Computers

What is a computer?

- Machine that accepts data, processes it and produces output
  - Anything with a CPU, mobile, phone, even fridge if it has a CPU
- Computer consists of:
  - Central processing unit (CPU)
  - Input/Output devices
  - Dynamic memory
  - Secondary memory
- This allows it to run software and applications

Data Representation

- Computer memory is made of components that are either on or off (binary)
  - 1 binary digit = 1 bit
  - 8 bits = 1 byte
- Binary can be used for a number of things and is done differently by different programs
  - Decimal - $42_{10} = 4*10^1 + 2*10^0$
    - The word "Python" would be - (81, 121, 116, 104, 111, 110 in ASCII(decimal))
  - Binary - $42_{10} = 1*2^5+0*2^4+1*2^3+0*2^2+1*2^1+0*2^0 = 101010_2$ (all to the base 2, not just the final 0)
    - The word "python" would be converted from decimal to - 01010000|01111001|01110100|01101000|01101111|011011 in binary

Operating Systems

- Applications - what the user interacts with, and feeds information to the operating system
- Operating system - a middle man between the applications and the hardware
- Hardware - The actual computer itself

Programming Languages

- Compiled languages
  - Advantages:
    - Very fast
    - Optimised compilers
    - Optimised libraries
  - Disadvantages:
    - Difficult to learn
    - Less readable code
    - Non- interactive
    - Verbose syntax (Gives you all the information, rather than just what you need)
  - Includes: Java, C, C++
- Scripting languages
  - Advantages:
    - Very rich libraries
    - Friendly development environment
    - Graphical visual interface
  - Disadvantages:
    - Sometimes expensive
    - Slower performance
  - Matlab, R, Python

Software Development

- Development process:
  1. Problem definition:
  - Requirements
    * A "top level" general English description of the problem/goal
  - Input and output data
    * Input - is the data required to solve the problem/achieve the goal
    * Output - is the results you want to get at the end
  - Processing needed
    * What need to happen to get that result
  - Maybe even how to test
  - Also need to consider security and performance requirements
  2. Design:
  - Think/consult/revise
    * How to solve the problem
    * Consider options:
      · Platform/operating system
      · Programming language
      · Libraries (existing code)
      · Structure
      · Take into consideration what constraints there are
  - Typically UML and descriptive text
  3. Specification:
  - Algorithm design
    * An algorithm is a set of detailed, unambiguous, ordered steps specifying a solution to a problem
    * Steps must be states precisely
    * Enter at the start, exit at the bottom
    * English description independent of any programming language
    * Non-trivial problem will need several stages of refinement
    * Must be desk-checked for correctness
  - More detail on how to implement
    * How to "break-up" the program
    * What libraries to use and patterns
    * If necessary, how your program will talk with other programs
  4. Implementation
  - Translate design into high level programming language/turn into code
    * Might need more than one language
    * Add comments/documentation to make is easier to understand
    * Break code into modules so it's not a huge file doing everything
    * Check if you've covered everything
  5. Execution
  - Conversion to machine instructions
    * To execute in python, we type: python test.py (or python3 test.py)
  - Python converts as code is executed (interpreted) one line at a time

6. Testing
   - Execute code and test that is actually does the required job
   - Test cases and harness
   - Finding errors and bugs:
     * Syntax errors -
       · Are quivalent to speling and punctuation mistakes
       · Python will stop execution when it interprets a syntax error - the line of code will not execute
     * Runtime errors -
       · Indicates something exceptional has gone wrong (divided by 0, things like that)
       · Python will execute the line and then exit with an error message
     * Semantic errors -
       · Harder to find, the code will run, but your results will be incorrect
   - Best to test as you go
   - Test every possible path
   - Testing can prove the existence of errors - not absense
- You might go back and forth between stages

Scrum Process

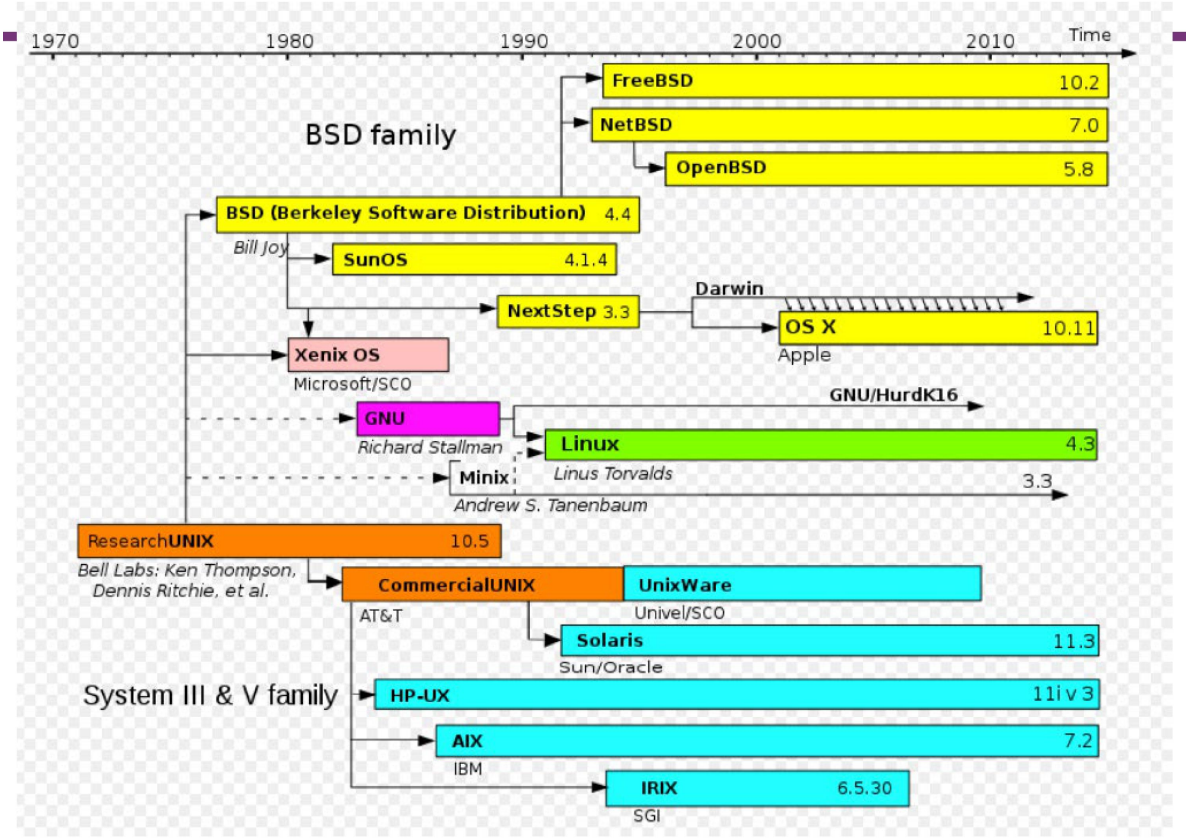
- Scrums are a way of breaking up development into smaller chunks to fit a timeline
  - Usually a scrum timeline is for 30 days
    * Sprint backlog -
      · Features to be covered in the sprint
    * Backlog -
      · Items expanded by the team
    * Product backlog -
      · Prioritised product features desired by the client
  - The 30 day timeline is further broken down into 24 hour periods
    * There is a 15 minute daily scrum meeting where team members describe:
      · What I've done since the last scrum meeting
      · What I plan to do before the next
      · Issues I have that I need help to resolve

Unix

- Unix is a general-purpose, time-sharing operating system
- Their philosophy is:
  - Write programs that do one thing and do it well
  - Write programs that work together
  - Write programs that handle text streams as a universal interface

Linux

- A variant of Unix
- Open source and free
- Created by Linus Torvalds
- A kernal (base operating system) with GNU OS utilities
- Unix and its subsidiaries:



Using Linux

- Uses a command line interface (CLI)
- Type commands at a prompt, press enter and the command will be processed
- Most commands follow the general form:
  - command[option(s)][filenames(s)]
- All case sensitive: cd $\neq$ CD $\neq$ Cd $\neq$ cD

Shells

- Shells are different versions of command-line interpreters
- Popular shells include:
  - bash - Bourne-Again Shell, default for most Linux systems (what we use)
  - sh - Bourne Shell, precurser to bash
  - csh - C Shell, with syntax that resembles the C programmin language syntax
  - tcsh - improved version of C Shell
  - ksh - Korn Shell

Files and Directories

- Important files:
  - /bin/capt(or cp, cd, etc) - Essential system programs
  - /dev/cdrom(or tty10, or lp2) - Files pointing to devices
  - /home/16167920 - User directory
  - /opt/anaconda/anaconda3/bin/python3 - Python3
  - /usr/bin(or local) - Programs for users

Piping and Grep

- Pipes are a Unix feature which allows you to connect several commands together
- The output of one command becomes the input to the next

Python

- Python was created by Guido van Rossum
- Free and open source, with on-going development from the Python Software Foundation
- Packages are written and shared by the community and PyPI
- We are using Python version 3
  - If you type in python, it will tell you what version you have, just make sure you type in quit() after because it puts you in the interpreter

Variables

- You don't need to tell Python which type of variable it is, it will work it out for itself
- Assigning types of variables:
  - String - "" holds one or more characters
  - Integer - just a number with no decimal
  - Float - a number with a decimal
  - Complex - use j to represent an complex number
  - Boolean - True or False (must be capitalised)

Boolean

- True or False
  - Also equal to False:
    - 0, 0,0, 0j - zero numeric values
    - '' or "" - empty strings
    - None - the null value
  - Everything else is True (or not False)

Expressions

- Expressions use variables and values for calculations
- If there is a combination of types in the calculation, Python will adjust the type, or give an error

Control Structures

- Used to repeat code, or choose whether to run part of the code
- Allows for decision making and repetition
- Main control structures in Python are:
  - **if/elif(else if)/else**
    - !!! The code within the if/elif/else block needs to be indented with 4 spaces
    - Multiple elif's can be used in sequence
  - **while**
  - **for**
- One entry, one exit
  - DO NOT USE BREAK OR CONTINUE, you will lose marks
- Control structures can be nested inside each other
  - Indent with 4 more spaces for each level of nesting
- Infinite loops will go on forever if they never reach the end condition