第 6 讲：The Programming Languages of OS

第一节：Introduction

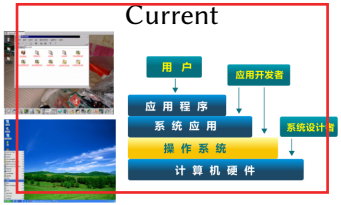
陈渝

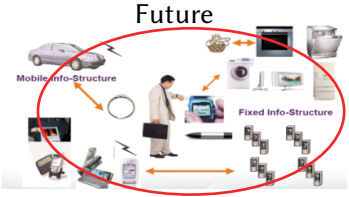
清华大学计算机系

yuchen@tsinghua.edu.cn

2020 年 3 月 22 日

Introduction



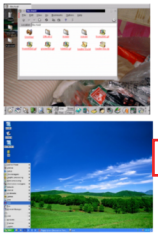


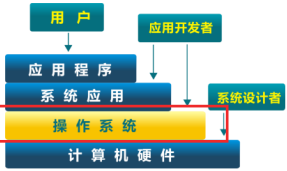
Some questions

- Is Programming Language important for OS?
- Is C the best language for OS?
- Which parts of language affect OS deeply?
- Other than languages, Which components affect the development of OS?
- Why do we need/needn't to change the language for OS?
- Other Langs(such as Rust, Go, etc.) based OS will be the future?

Introduction

Current





Future

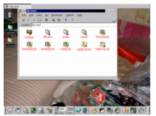
What is OS

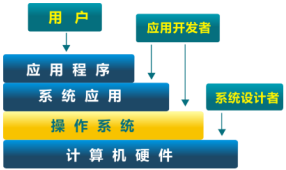
An operating system (OS) is system software that manages computer hardware, software resources, and provides common services for computer programs. [wikipedia]

What is the Programming Languages?

Introduction

Current





Future

What is Programming Languages

A programming language is a formal language, which comprises a set of instructions that produce various kinds of output. Programming languages are used in computer programming to implement algorithms. [wikipedia]

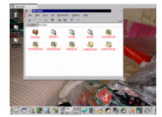
What is the Programming Languages for OS?

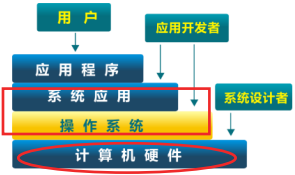
Introduction

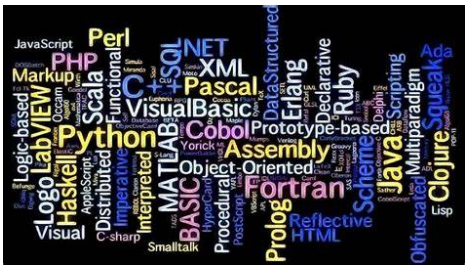
TIOBE Index for August 2019

Aug 2019	Aug 2018	Change	Programming Language	Ratings	Change
1	1		Java	16.028%	-0.85%
2	2		C	15.154%	+0.19%
3	4	↑	Python	10.020%	+3.03%
4	3	↓	C++	6.057%	-1.41%
5	6	↑	C#	3.842%	+0.30%
6	5	↓	Visual Basic .NET	3.695%	-1.07%
7	8	↑	JavaScript	2.258%	-0.15%
8	7	↓	PHP	2.075%	-0.85%
9	14	⇈	Objective-C	1.690%	+0.33%
10	9	↓	SQL	1.625%	-0.69%

Introduction





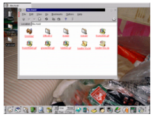


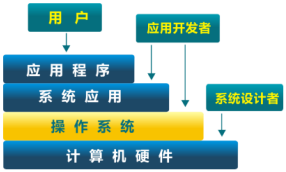
What is System programming language

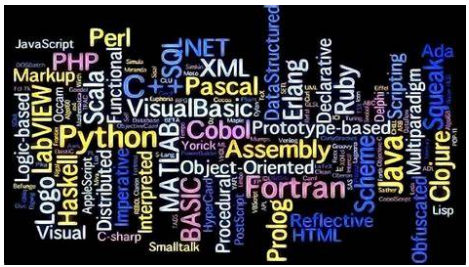
A system programming language is a programming language used for system programming; such languages are designed for writing system software, which usually requires different development approaches when compared with application software. Edsger Dijkstra refers to these language as Machine Oriented High Order Languages. [wikipedia]

- General-purpose programming languages (Java, Pascal, etc.) tend to focus on generic features. This generic quality typically comes at the cost of ~~denying direct access to the machine's internal workings,~~ and this often has negative effects on performance.

Introduction





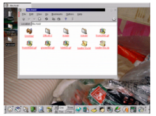


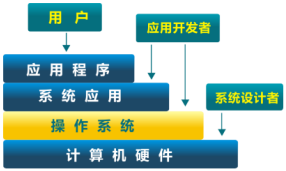
What is System programming language

A system programming language is a programming language used for system programming; such languages are designed for writing system software, which usually requires different development approaches when compared with application software. [wikipedia]

- System languages are for performance and ease of access to the underlying hardware while still providing high-level programming concepts like structured programming. Examples include Executive Systems Problem Oriented Language (ESPOL), which is similar to ALGOL in syntax but tuned to their respective platforms.

Introduction





What is System programming language

A system programming language is a programming language used for system programming; such languages are designed for writing system software, which usually requires different development approaches when compared with application software. [wikipedia]

- Some languages straddle the system and application domains, bridging the gap between these uses. The canonical example is C, which is used widely for both system and application programming. Some modern languages also do this such as Rust and Swift.

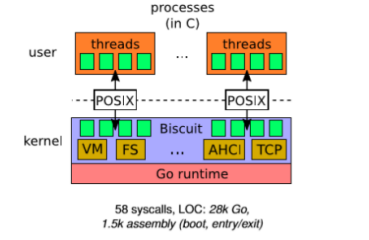
Introduction

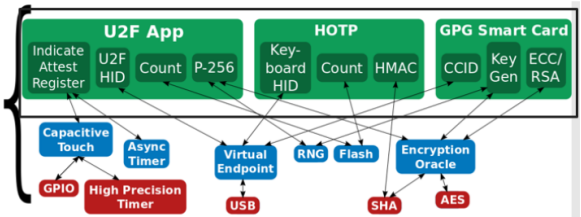
Major System programming languages

[hide] Language	Originator	Birth date	Influenced by	Used for
ESPOL	Burroughs Corporation	1961	ALGOL 60	MCP
PL/I	IBM, SHARE	1964	ALGOL, FORTRAN, some COBOL	Multics
PL360	Niklaus Wirth	1968	ALGOL 60	ALGOL W
C	Dennis Ritchie	1969	BCPL	Most operating system kernels, including Unix-like systems
PL/S	IBM	196x	PL/I	OS/360
BLISS	Carnegie Mellon University	1970	ALGOL-PL/I[5]	VMS (portions)
PL/8	IBM	197x	PL/I	AIX
PL-6	Honeywell, Inc.	197x	PL/I	CP-6
SYMPL	CDC	197x	JOVIAL	NOS subsystems, most compilers, FSE editor
C++	Bjarne Stroustrup	1979	C, Simula	C++ Applications[6]
Ada	Jean Ichbiah, S. Tucker Taft	1983	ALGOL 68, Pascal, C++, Java, Eiffel	Embedded systems, OS kernels, compilers, games, simulations, CubeSat, air traffic control, avionics
D	Digital Mars	2001	C++	Multiple domains
Nim	Andreas Rumpf	2006	Python), Ada, Lisp, Oberon, C++, Modula-3, Object Pascal	Games, compilers, OS kernels, app development, embedded systems, etc.
Rust	Mozilla Research[7]	2010	C++, Haskell, Erlang, Ruby	Servo, Redox OS
Swift	Apple Inc.	2014	C, Objective-C, D, Rust	macOS, iOS app development [a]

Introduction

Some OS based on Non-C language



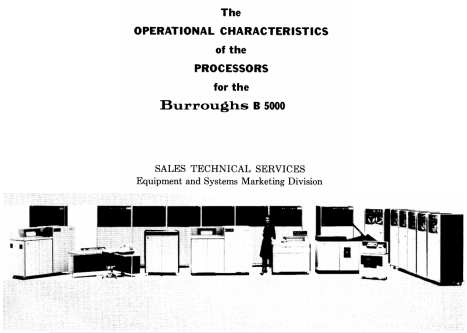


go-lang based Biscuit OS, MIT, OSDI'2018

rust-lang based tock OS, Stanford, SOSP'2017

GOLANG

History – MCP & ESPOL




```
BEGIN
  FILE F(KIND=REMOTE);
  EBCDIC ARRAY E[0:11];
  REPLACE E BY "HELLO WORLD!";
  WRITE(F, *, E);
END.
```

- The MCP (Master Control Program), the first OS written exclusively in a high-level language (HLL), is the proprietary operating system of the Burroughs small, medium and large systems.
- MCP provides virtual memory, file system with hierarchical directory structures, processes are called ”Jobs” and ”Tasks”, software components and libraries.
- Originally written in 1961 in ESPOL (Executive Systems Programming Language), which itself was an extension of Burroughs Extended ALGOL,

History – MCP & ESPOL

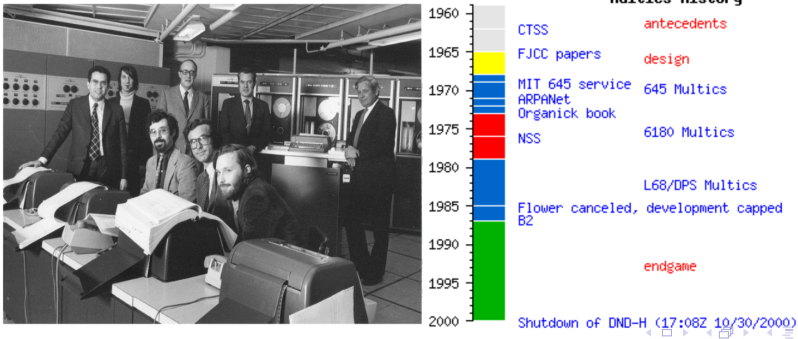
ESPOL (Executive Systems Programming Language), extension of Burroughs Extended ALGOL.

```
2  ⟨program⟩ ::= ⟨block⟩ . ⟨space⟩

2  ⟨block⟩ ::= ⟨unlabeled block⟩

1  ⟨unlabeled block⟩ ::= ⟨block head⟩; ⟨compound tail⟩

1  ⟨block head⟩ ::= BEGIN ⟨declaration⟩ | ⟨block head⟩;
                      ⟨declaration⟩

1  ⟨compound tail⟩ ::= ⟨statement⟩ END | ⟨statement⟩;
                       ⟨compound tail⟩

2  ⟨declaration⟩ ::= ⟨variable declaration⟩ | ⟨label declaration⟩ |
                     ⟨switch declaration⟩ |
                     ⟨procedure declaration⟩ |
                     ⟨stream procedure declaration⟩ |
                     ⟨subroutine declaration⟩ |
                     ⟨define declaration⟩ |
                     ⟨forward reference declaration⟩

2  ⟨statement⟩ ::= ⟨unconditional statement⟩ |
                   ⟨conditional statement⟩ |
                   ⟨iterative statement⟩ |
                   ⟨in-line character mode statement⟩

2  ⟨unconditional statement⟩ ::= ⟨compound statement⟩ |
                                 ⟨basic statement⟩
```

History – MULTICS & PL/I

MULTICS OS & PL/I language

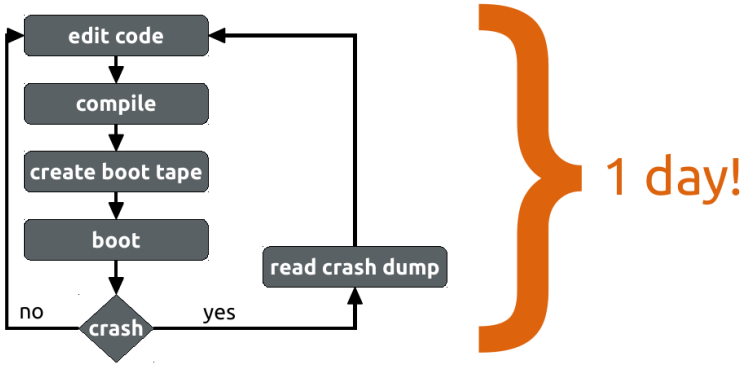
- Multics (Multiplexed Information and Computing Service) is an influential early time-sharing operating system. Virtually all modern operating systems were heavily influenced by Multics –often through Unix.
- PL/I (Programming Language One, and sometimes written PL/1)is a procedural, imperative computer programming language developed by IBM.



History – MULTICS & PL/I

Development on (heavily loaded) CTSS or MULTICS

- MIT: Prior pioneering work
- Bell Labs: Engineering, Business management, Communications
- General Electric: Hardware



History – MULTICS & PL/I

The Choice of PL/I

- PL/I (Programming Language One, and sometimes written PL/1)is a procedural, imperative computer programming language developed by IBM.
- In 1964, when Project MAC at MIT sought to build a successor to their Compatible Time-sharing System (CTSS), they selected the language (PL/I) before writing any code.
- The decision to use PL/I in Multics was seen by its creators as a great strength. But that the compiler was unavailable for so long (and when was available, performed poorly) was a nearly-fatal weakness.
- The dynamic linking and paging facilities of the Multics environment have the effect of making available in virtual storage only those specific pages of those particular procedures which are referenced during an execution of the compiler.

Multics Logo
https://multicians.org/pl1.html
https://multicians.org/pl1-raf.html
http://teampli.net/plisprg.html

History – UNIX & C

The birth of Unix

- Bell Labs pulled out of the Multics project in 1969.
- A researcher formerly on the Multics effort, Ken Thompson, implemented a new operating system(was implemented entirely in assembly) for the PDP-7.
- The system was later ported to the PDP-11/20, where it was named Unix —a play on “eunuchs” and a contrast to the top- down complexity of Multics.

History –Unix & high-level languages

B –> C

- The interpreted language B (a BCPL derivative), was present in Unix, but only used for auxiliary functionality, e.g. the assembler.
- Some of the B that was in use in Unix was replaced with assembly for reasons of performance.
- Dennis Ritchie and Thompson developed a B-inspired language focused on better abstracting the machine, naming it "C".
- Perhaps contrary to myth, C and Unix were not born at the same instant —they are siblings, not twins!

History –Unix & high-level languages

The C revolution

- C is rightfully called “portable assembly”: it is designed to closely match the abstraction of the machine itself.
- C features memory addressability at its core.
- Unlike PL/I, C grew as concrete needs arose.
- e.g., C organically adopted important facilities like macro processing through the C preprocessor.
- Standardization efforts came late and were contentious: C remains infamous for its undefined behaviors.

History –Unix & high-level languages

Operating systems in the 1980s

- As the minimal abstraction above the machine, C —despite its blemishes —proved to be an excellent fit for operating systems implementation
- With few exceptions, operating systems —Unix or otherwise —were implemented in C throughout the 1980s
- Other systems existed as research systems, but struggled to offer comparable performance to C-based systems

History –Unix & high-level languages

Operating systems in the 1990s

- In the 1990s, object oriented programming came into vogue, with languages like C++ and Java
- By the mid-1990s, C-based systems were thought to be relics
- ...but the systems putatively replacing them were rewrites —and suffered from rampant Second System Syndrome
- They were infamously late (e.g. Apple's Copland), infamously slow (e.g. Sun's Spring), or both (Taligent's Pink)
- Java-based operating systems like Sun's JavaOS fared no better; hard to interact with hardware without unsigned types.

History –Unix & high-level languages

Operating systems in the 2000s

- With the arrival of Linux, Unix enjoyed a resurgence, and C-based operating systems became deeply entrenched
- With only a few exceptions (e.g., Haiku), serious attempts at C++-based kernels withered
- At the same time, nonJava/non-C++ languages blossomed: first Ruby, and then Python and JavaScript
- These languages were focused on ease of development rather than ~~performance~~ and there appears to be no serious effort to implement an operating system in any of these

History –Unix & high-level languages

Systems software in the 2010s

- Systems programmers began pining for something different: the performance of C, but with more powerful constructs as enjoyed in other languages
- High-performance JavaScript runtimes allowed for a surprising use in node.js —but otherwise left much to be desired
- Bell Labs refugees at Google developed Go, which solves some problems, but with many idiosyncrasies
- Go, JavaScript and others are garbage collected, making interacting with C either impossible or excruciatingly slow
- Rust is a systems software programming language designed around safety, parallelism, and speed

History –Unix & high-level languages

Rust &OS in the 2010s

- First attempt at an operating system kernel in Rust seems to be Alex Light's Reenix, ca. 2015: a re-implementation of a teaching operating system in Rust as an undergrad thesis
- Since Reenix's first efforts, there have been quite a few small systems in Rust, e.g. Redox, Tifflin, Tock, intermezzOS, RustOS/QuiltOS, Rux, and Philipp Oppermann's Blog OS, and rcore.
- Some of these are teaching systems (intermezzOS, Blog OS), some are unikernels (QuiltOS) and/or targeted at IoT (Tock)

C based OS will dominate the development of OS?
Other Langs(such as Rust, Go, etc.) based OS will be the future?

References

- Is it time to rewrite the operating system in Rust? Bryan Cantrill, tech talk, 2018