Computers
Fundamentals of Programming

Grau en Ciència i Enginyeria de Dades

Xavier Martorell, Xavier Verdú

Facultat d'Informàtica de Barcelona (FIB)

Universitat Politècnica de Catalunya (UPC)

2019-2020 Q2

Creative Commons License

Table of Contents

Understanding your application

- While developing/maintaining code, it is extremely important!!
- Have in mind:
  - Goal of the application
  - Algorithms
  - Data structures
  - Which services it uses from the system
  - Structure of the source code
    - Directory structure
    - Header files
    - Source files
  - Structure of the binary files
    - When compiled

Understanding your application

- Example...  Geany git

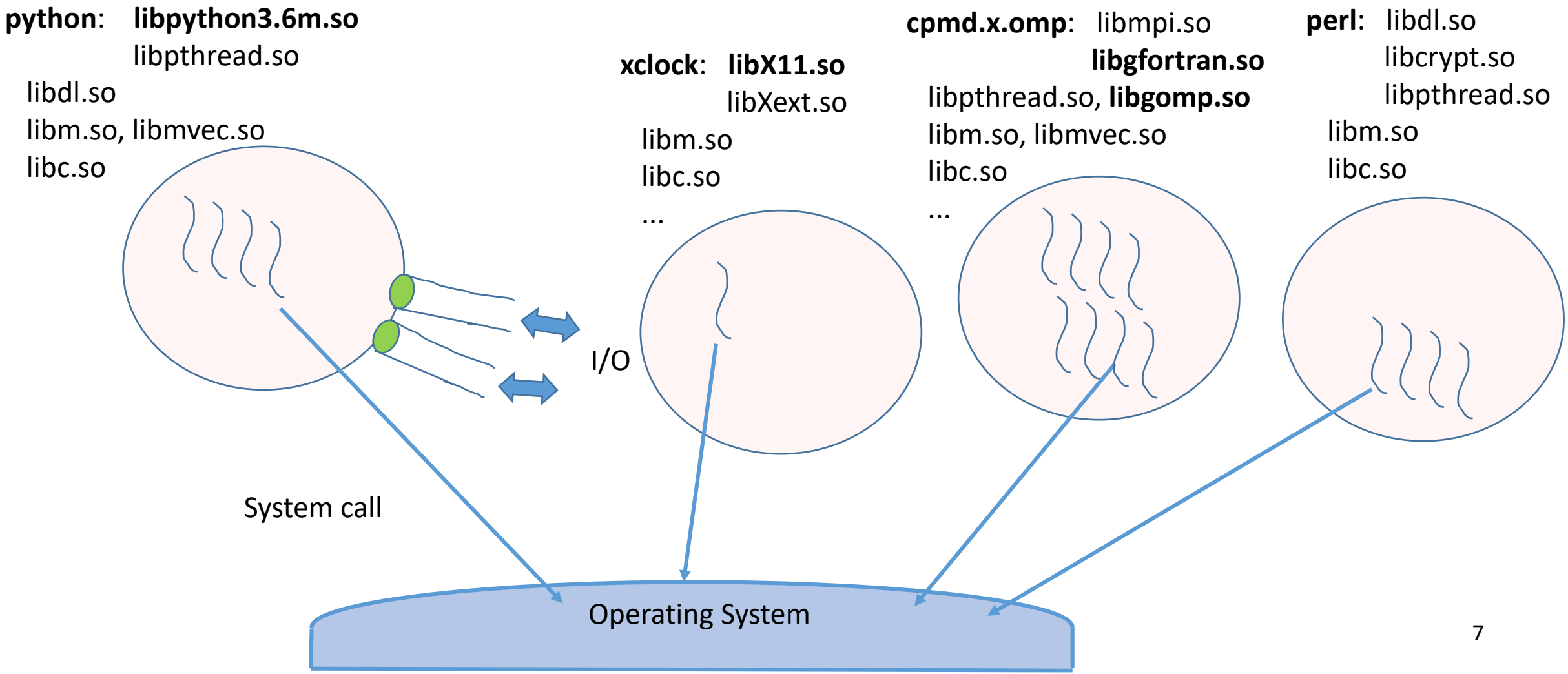
data	Theme improvements (#1382)
doc	manual: added documentation about replacement of 'untitled.ext' with ...
icons	icon: regenerate png/ico files based on the svg
m4	Update Scintilla to version 3.7.5 (#1503)
plugins	filebrowser: Don't change directory on project save
po	Small update of German translation
scintilla	Update Scintilla to version 3.7.5 (#1503)
scripts	Update Scintilla to version 3.7.5 (#1503)
src	Merge pull request #1748 from kugel-/msgwin-api
tests	bash may not found in the system (#1574)

Understanding your application

- autotools
- src
- include

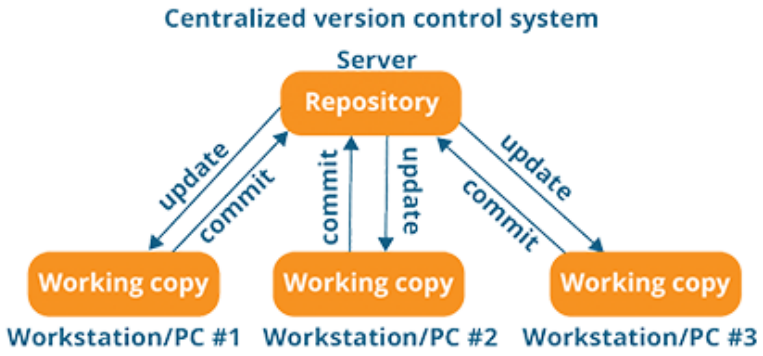
Makefile.am	Merge pull request #1095 from eht16/issue1076_win32_build_working_dir...
about.c	Remove a space (#1790)
about.h	Normalize use of header guards and extern "C" guards
app.h	Add utils_get_real_path() and use it
build.c	Work around a `-Wformat-overflow` warning
build.h	doxygen: various doxygen-related fixes in preparation for gtkdoc gene...
callbacks.c	Show status message on attempt to execute empty context action.
callbacks.h	Allow to set a keybinding for File->Properties
dialogs.c	Fix canceling keybinding overriding by discarding the dialog
dialogs.h	Protect private definitions by the GEANY_PRIVATE macro in headers
document.c	Add missing space in string. Fixes #1789
document.h	Added option to auto reload files changed on disk (#1246)
documentprivate.h	Add support for Keyed Data Lists for documents
editor.c	Remove some unused variables

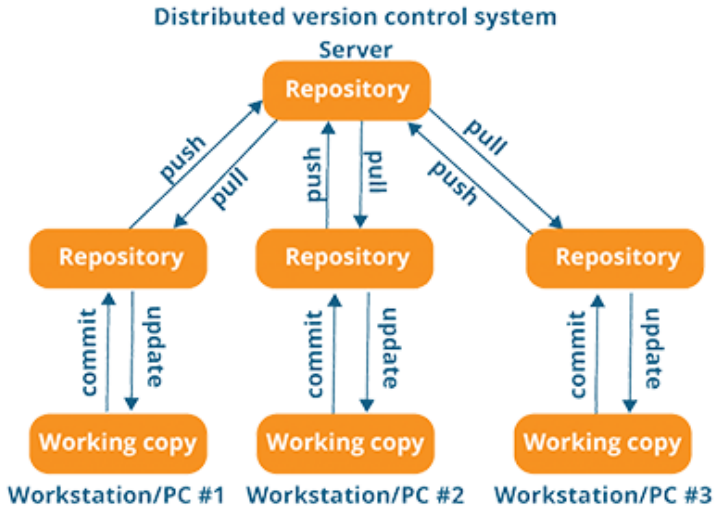
Library support according to language and OS



Code Version Management

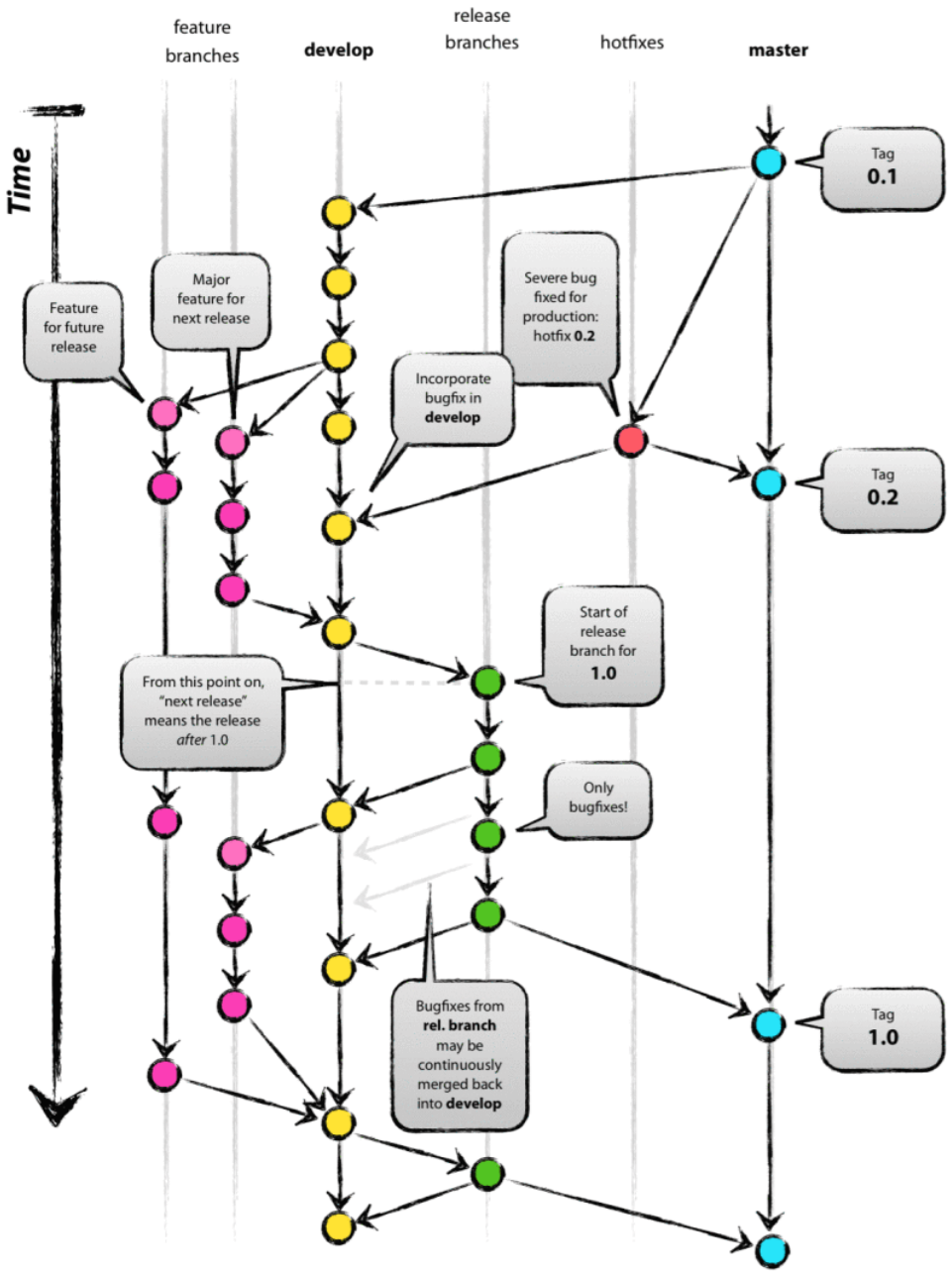
- Version Control tools
  - A set of tools that help to keep track of changes in code using a hierarchy of internal structures and files that help to manage different concurrent versions of code
    - Centralized Version Control System (e.g. cvs, svn): there is a single copy of the **repository** (i.e. all code versions)
    - Distributed Version Control System (e.g. **git**, Mercurial): there are multiple copies of the repository
  - There are web-accessible repositories where people/companyies manage code
    - E.g.: Github, Gitlab





https://www.atlassian.com/blog/software-teams/version-control-centralized-dvcs

Ex. Code Management



Compilation and code management toolchains

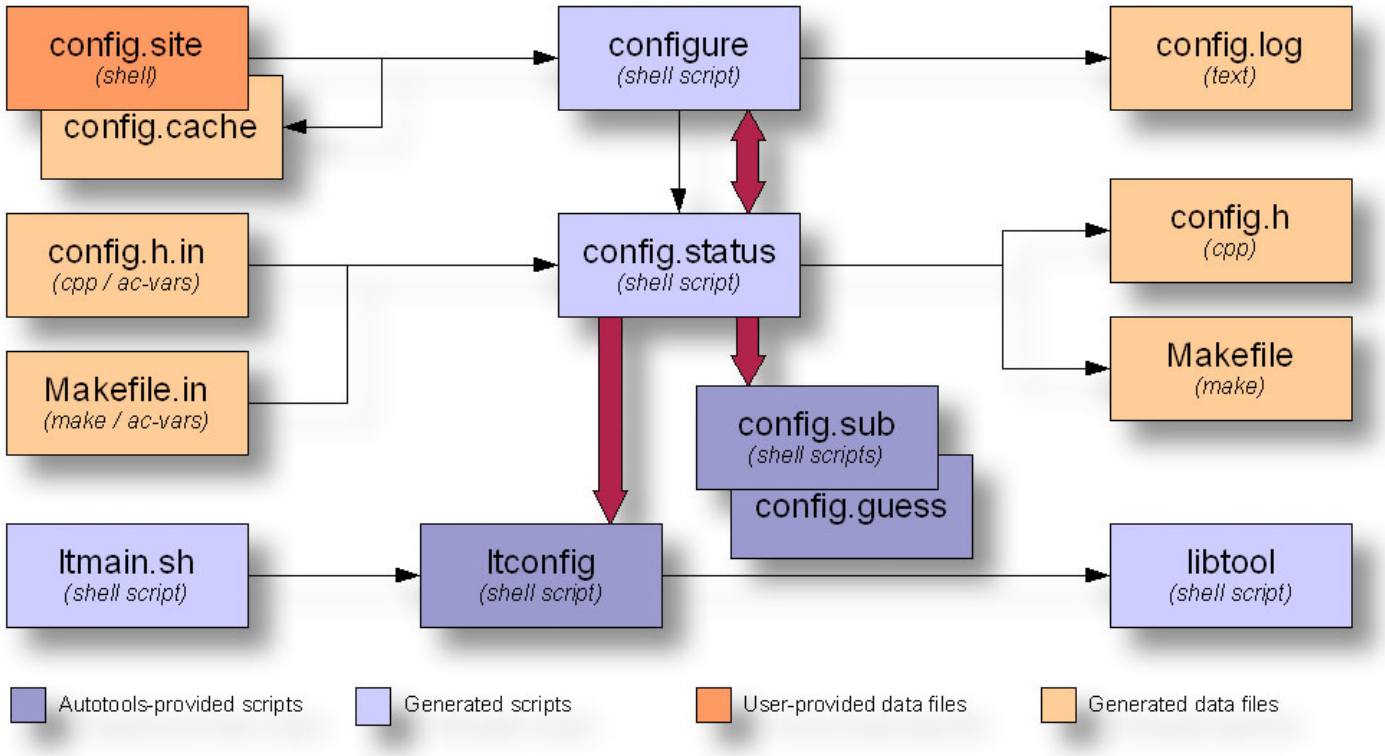
- Software Version Formats
  - (X.Y.Z)
    - X: Major changes, usually incompatible with previous versions;
    - Y: Minor changes, new functionalities added in a backwards-compatible manner;
      Z: Revision/Patch (bug fixing)
  - Odd-even System
    - Odd numbers for development and even numbers for stable releases

- Build Process Tools
  - A set of tools for software developers to create/distribute automatically buildable source code and make software packages portable
  - Autotools (by GNU) make it easier to suport portability, (build configuration) based on common build conventions (e.g. well known paths), and automate dependency tracking to create the makefile
    - autotools = autoconf + automake + libtool + ...
  - Cmake is the next generation of autotools

Example of compile and install process

(terminal point of view)

```
#> Download source code
...
#> ./configure
```



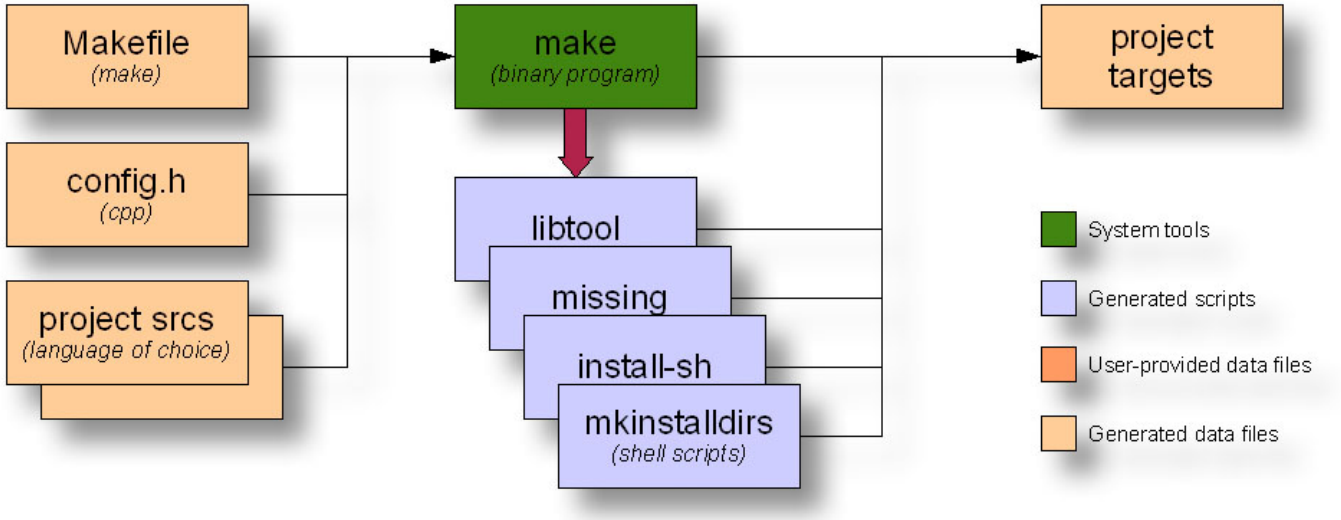
Configure data flow diagram

http://freesoftwaremagazine.com/articles/brief_introduction_to_gnu_autotools/

Example of automated build process

(terminal point of view)

```
#> Download source code
…
#> ./configure
#> make all
#> make install
```



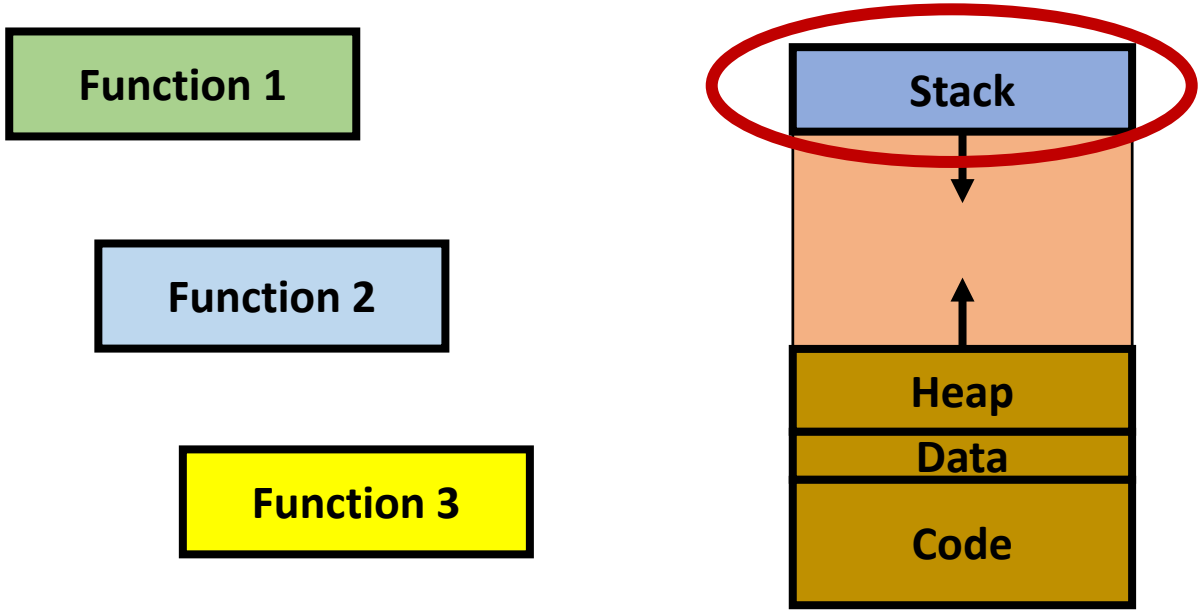
Make data flow diagram

http://freesoftwaremagazine.com/articles/brief_introduction_to_gnu_autotools/

Table of Contents

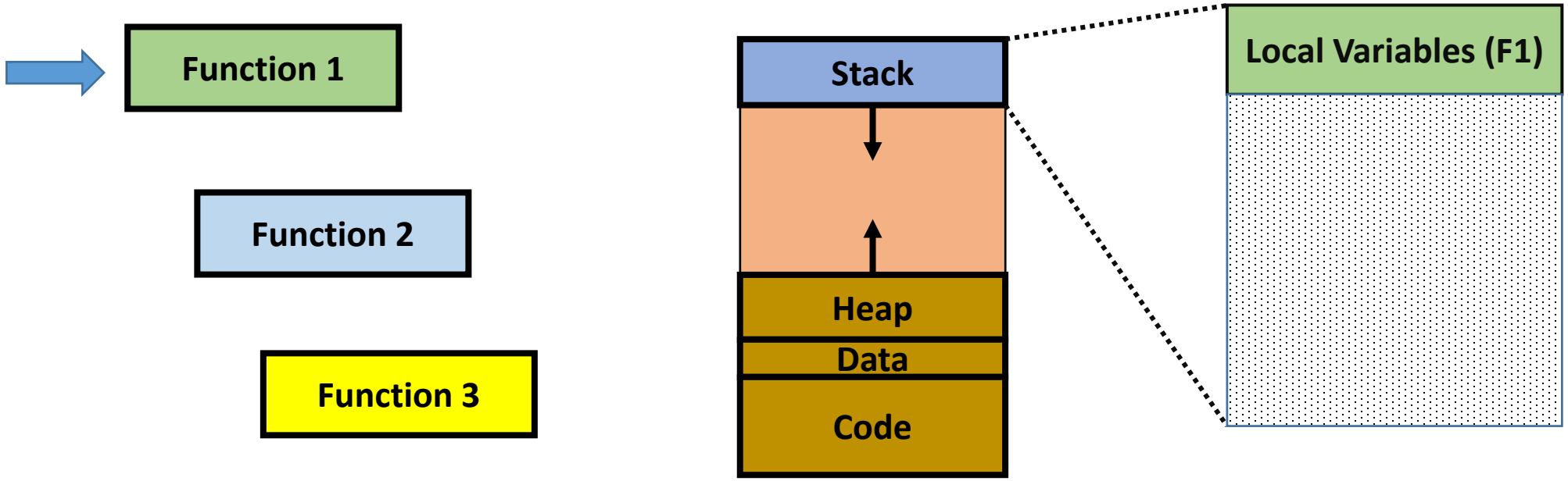
Calling conventions

- The calling conventions define the standard to call a function
  - It mainly defines how to call the function and to manage registers and stack for
    - arguments, local variables, return value, return address



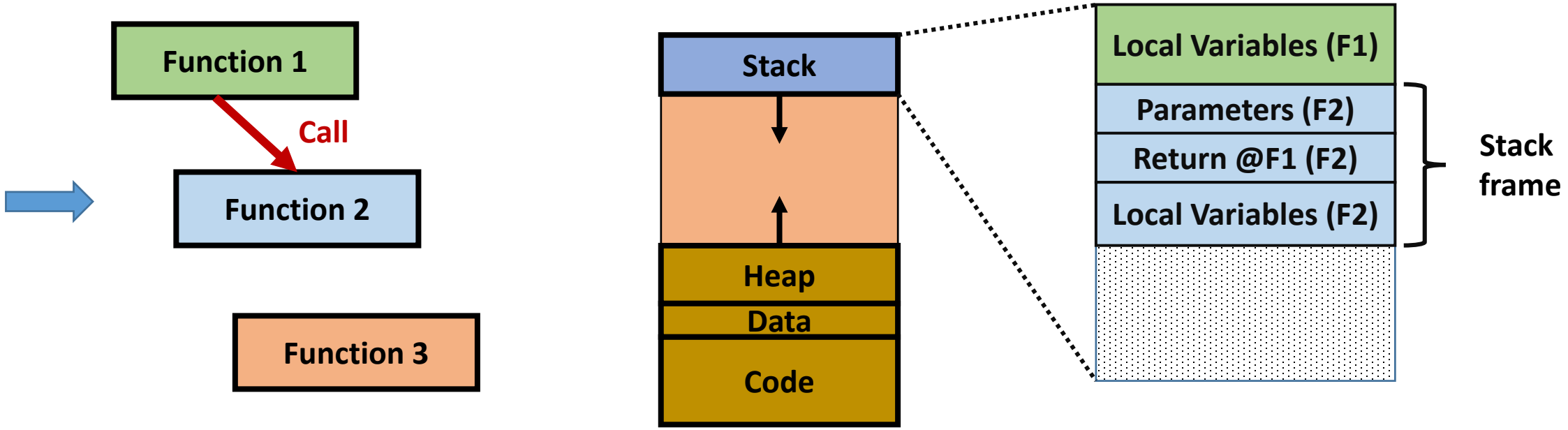
Calling conventions

- The calling conventions define the standard to call a function
  - It mainly defines how to call the function and to manage registers and stack for
    - arguments, local variables, return value, return address



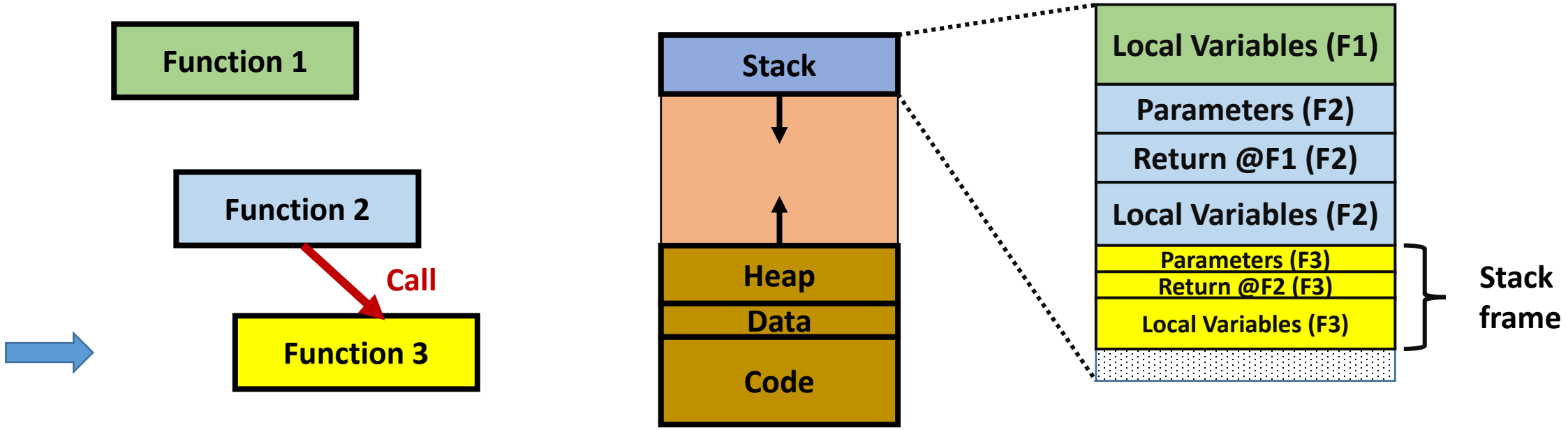
Calling conventions

- The calling conventions define the standard to call a function
  - It mainly defines how to call the function and to manage registers and stack for
    - arguments, local variables, return value, return address
  - **Stack frame** is the stack's active part of the current function



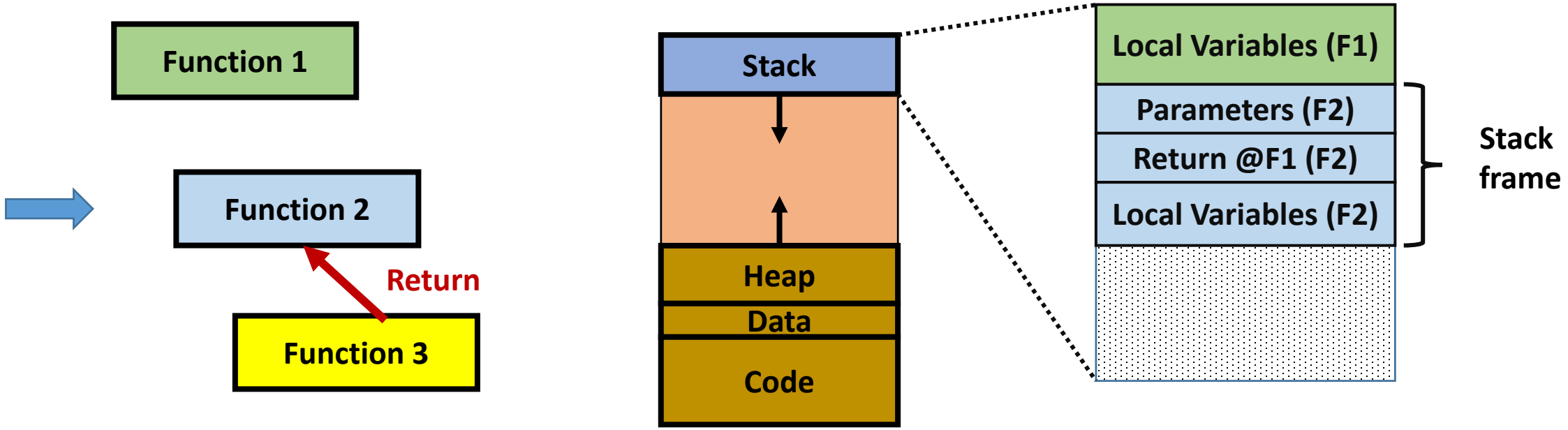
Calling conventions

- The calling conventions define the standard to call a function
  - It mainly defines how to call the function and to manage registers and stack for
    - arguments, local variables, return value, return address
  - **Stack frame** is the stack's active part of the current function



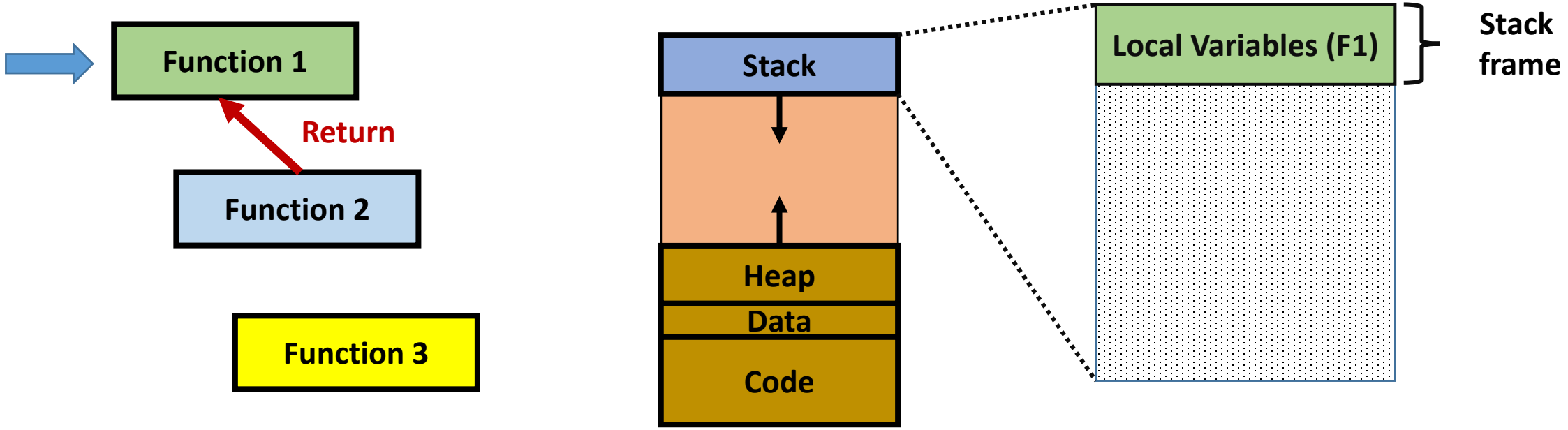
Calling conventions

- The calling conventions define the standard to call a function
  - It mainly defines how to call the function and to manage registers and stack for
    - arguments, local variables, return value, return address
  - **Stack frame** is the stack's active part of the current function



Calling conventions

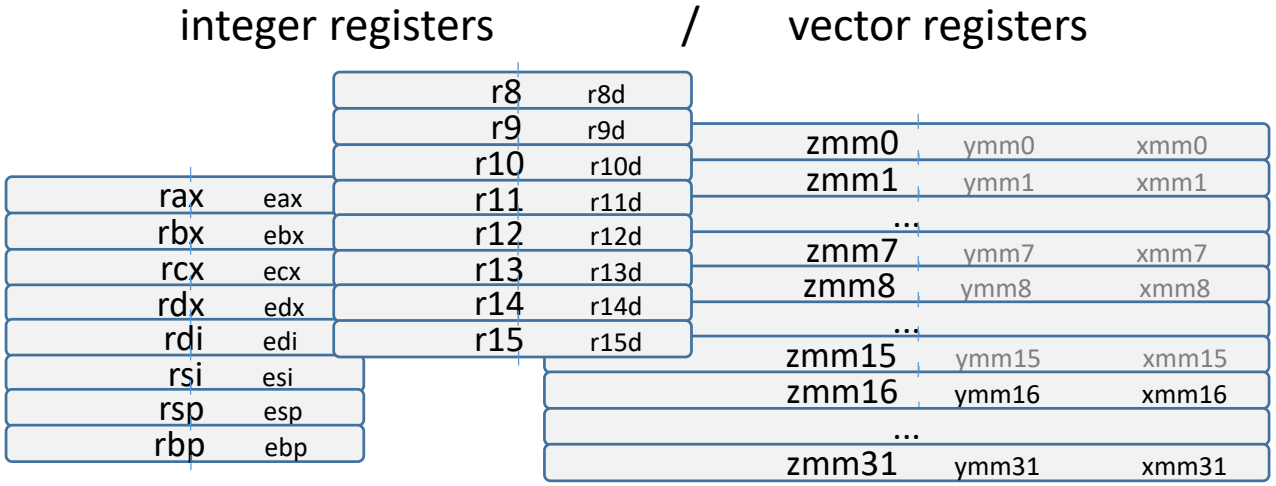
- The calling conventions define the standard to call a function
  - It mainly defines how to call the function and to manage registers and stack for
    - arguments, local variables, return value, return address
  - **Stack frame** is the stack's active part of the current function



Calling conventions

- There are several calling conventions depending on: the programming language, compiler and processor architecture

- For example, to return a value from a function, depending on the architecture:
  - x86 uses eax register
  - x86_64 uses rax register
  - ARM uses r0 register



- There is a summary table in
  https://wiki.osdev.org/Calling_Conventions

  https://en.wikipedia.org/wiki/X86_calling_conventions

Calling conventions of C/C++

```
int function (int A, int B)
{
    int res;
    …
    return res;
}
```

- C/C++ present several well known calling conventions, such as:
  - **Cdecl** (stands for C declaration) that is the default C calling convention
    - Arguments are passed from right-to-left. Return value through eax/rax register
    - The calling function cleans the stack
    - Name decoration: underscore as the prefix of the function name: *_function*
  - **Stdcall** (also known as "WINAPI") that is the standard convention for Win32 apps (by Microsoft)
    - Arguments are passed from right-to-left. Return value through eax/rax register
    - The called function cleans the stack **(no variable-length argument lists)**
    - Name decoration: underscore as the prefix of the function name and "@" followed by the number of bytes in the argument list: *_function@8*

- **Name Decoration** (a.k.a. name-mangling) is an internal encoded string generated at compile-time to support the linker, especially to resolve unique function names

Automated Software Deployment

- Everytime there are new code developments, the mainline code has to be updated in a secure way. That is, the code needs to be tested
  - There are software projects that need to automate this task

- There are different procedures depending on the target
  - **Continuous Integration:** automates built and test of new code. The main goal is to integrate it to the mainline code. Thus, it has to be tested before and after the integration
  - **Continuous Delivery:** automates a new software release. The main target is to perform user acceptance tests
  - **Continuous Deployment:** is a step up, since it deploys the new release to production stage. That is, to the end-user

- There are tools that integrates multiple tools to perform these tasks
  - E.g. Jenkins

Bibliography

- Microsoft Developer Network
  - Calling conventions
    - https://docs.microsoft.com/en-us/cpp/cpp/calling-conventions?view=vs-2019
- OS Dev (.org)
  - Calling conventions
    - https://wiki.osdev.org/Calling_Conventions
- Wikibooks
  - X86 calling conventions
    - https://en.wikipedia.org/wiki/X86_calling_conventions
  - X86 disassembly/calling conventions
    - https://en.wikibooks.org/wiki/X86_Disassembly/Calling_Conventions
- Continuous management of software releases
  - Atlassian Wiki
    - https://www.atlassian.com/continuous-delivery/principles/continuous-integration-vs-delivery-vs-deployment