동적 환경변수 관리 도구를 활용한 OpenFOAM 사용환경 구축 및 OpenFOAM 개발을 위한 IDE 환경 구축

삼성중공업
해양부유체연구파트
연성모

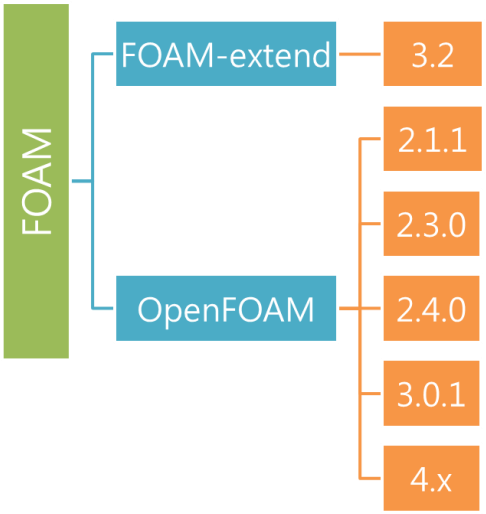
2016.09.29-30

Overview

- 동적환경 변수 관리도구를 활용한 OpenFOAM 사용환경 구축
  - Background
    - 사용자 관점에서의 요구사항
  - Environment modules 소개
  - OpenFOAM 적용 예
  - 요약 및 결론
- OpenFOAM 개발을 위한 IDE 환경 구축
  - Background
  - Linux에서의 intellisense 현주소
    - Gcc의 한계
    - Clang 소개
  - IDE 환경 구현
    - Plugin 설치 절차
    - 설정파일 예
    - VIM/Emacs 코드 작성 예
  - 요약 및 결론

동적 환경변수 관리 도구를 활용한 OpenFOAM 사용환경 구축

- OpenFOAM 버전 관리의 필요성
  - OpenFOAM version : 2.3.x, 2.4.x, 3.0.x, 4.x and so on
  - FOAM-extend version : 3.2, 4.0 and so on
- 전통적인(?) alias를 이용한 관리의 문제점
  - Dirty .bashrc
    - 버전이 많아질 수록 bashrc 파일이 지저분해짐
  - PATH, LD_LIBRARY_PATH 처리 불가
    - .bashrc에서 처리해야함
  - 다양한 Compiler + MPI 조합의 building 환경 고려 못함
    - 로그인시마다 별도의 .bashrc를 재호출
    - MPI의 경우 compiler dependent code 생성, 호환성 없음
  - 환경변수 오염
    - 서로 다른 버전의 OpenFOAM을 단일 shell에서 실행시킬 수 없음
      - wmUNSET 명령으로 환경변수 해제가 가능
        - » **별도의 script가 필요함**
        - » **FOAM-extend에서는 지원 안함**
    - **버전, 프로그램 종류에 관계없이 일관된/단순한 방법이 필요**
    - 다른 버전 사용시 현재 사용하는 terminal 활용못함



사용자 관점에서의 요구사항

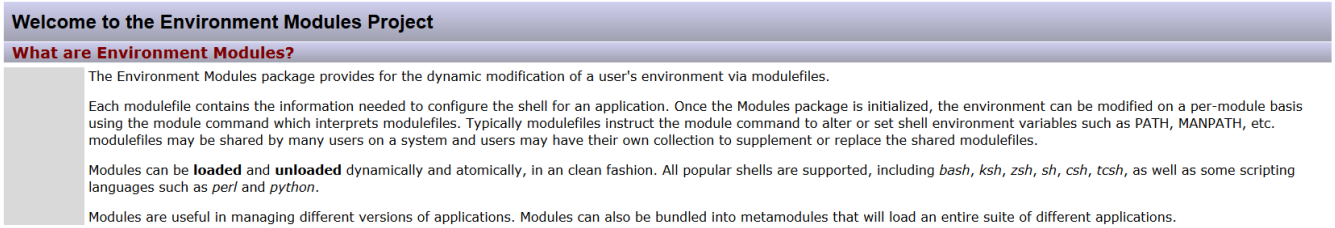
- 시스템 관리자
  - **일반유저에게 통일된 사용환경 제공**
    - 잘못된 환경변수 설정으로 인한 오류 방지
    - 오류 리포트 감소
    - **일반유저에게 복잡한 환경변수 설정내용을 감추고 interface만 제공**
  - 새로운 package 및 기능 추가가 용이
  - RHEL 환경에서 package 설치/관리 문제 해결
    - Old gcc version 4.4 (classic c++)
      - Gcc 4.8 부터 Modern c++ 지원 (C++11/14/17 등)
      - Intel Parallel Studio 2016 Update 3부터 c++11 완벽지원
      - OpenFOAM 4.x부터 c++11기반으로 작성
- 일반 유저
  - 사용 도구에 맞는 간편한 환경 설정
    - I am not a Linux expert
    - 직관적인 환경변수 접근방법 제공
  - OpenFOAM
    - Switching between versions

```
---------- /applic/Modules/2014_10/applications ----------
applic/amber-10       applic/lammps-10Aug15   applic/python-2.7.8
applic/espresso-5.4.0 applic/lammps-5Sep14    applic/qchem-3.2.0.3
applic/g09-b01        applic/namd-2.9         applic/root-6.04.02
applic/g09-d01        applic/ncview-2.1.1     applic/siesta-3.2-p1-5
applic/gromacs-4.6.7  applic/octopus-4.1.2    applic/siesta-4.0
applic/gromacs-5.0.6  applic/petsc-3.5.2      applic/totalview-8.8.0-2

---------- /applic/Modules/2014_10/compilers ----------
compiler/gcc-4.1.2 compiler/gcc-4.9.3 compiler/intel-2013 compiler/pgi-2014
compiler/gcc-4.4.6 compiler/intel-11.1 compiler/intel-2015 compiler/pgi-9.0-4

---------- /applic/Modules/2014_10/libraries ----------
applic/grads-2.0.a9 applic/hdf5       applic/lapack-3.5.0 applic/netcdf4
applic/hdf4         applic/hdf5-1.8.13 applic/ncarg-4.4.2  applic/netcdf4-4.1.3
applic/hdf4-4.2.10  applic/hdf5-1.8.7  applic/ncarg-5.2.1
applic/hdf4-4.2.6   applic/lapack-3.2.2 applic/ncarg-6.0.0

---------- /applic/Modules/2014_10/mpi ----------
mpi/intelmpi-2015 mpi/mvapich2-2.0 mpi/openmpi-1.4.2 mpi/openmpi-1.8.2
mpi/mvapich2-1.4  mpi/mvapich2-2.1 mpi/openmpi-1.4.3 mpi/openmpi-1.8.5
mpi/mvapich2-1.5  mpi/openmpi-1.3.3 mpi/openmpi-1.6.3

---------- /applic/Modules/2014_10/libraries_using_mpi ----------
applic/fftw-2.1.5        applic/fftw-3.2.1
applic/fftw-2.1.5-single applic/fftw-3.3.4

---------- /applic/Modules/2014_10/test ----------
intel-2016          intel_cluster_studio-2013
```

KISTI Tachyon II modules

Environment modules 적용

Environment modules



Welcome to the Environment Modules Project

What are Environment Modules?

The Environment Modules package provides for the dynamic modification of a user's environment via modulefiles.

Each modulefile contains the information needed to configure the shell for an application. Once the Modules package is initialized, the environment can be modified on a per-module basis using the module command which interprets modulefiles. Typically modulefiles instruct the module command to alter or set shell environment variables such as PATH, MANPATH, etc. modulefiles may be shared by many users on a system and users may have their own collection to supplement or replace the shared modulefiles.

Modules can be **loaded** and **unloaded** dynamically and atomically, in an clean fashion. All popular shells are supported, including *bash*, *ksh*, *zsh*, *sh*, *csh*, *tcsh*, as well as some scripting languages such as *perl* and *python*.

Modules are useful in managing different versions of applications. Modules can also be bundled into metamodules that will load an entire suite of different applications.

- http://modules.sourceforge.net/
- 환경변수의 동적관리를 위한 Tcl 기반 관리도구
- 대부분의 HPC 환경에서 environment modules 기반으로 사용자 환경 제공
  - KISTI Tachyon II
  - TACC Stampede
  - ONR HPCs
- Module 파일
  - 그 자체로 Tcl source code
    - 파일 내에서 조건부 환경변수 생성/관리 가능
    - package의 버전 및 compiler에 따른 환경변수 변경을 관리 가능함
- Package의 의존관계를 명시할 수 있음
  - 사용자 실수로 인한 오작동 방지
- Job 제출시 현재 shell 환경변수에 관계없이 OpenFoam version 호출이 가능해짐

Environment modules

- 사용가능한 환경 module 확인 및 현재 사용중인 module 확인 가능
  - module avail
  - module list
- 새로운 환경 변수 등록 및 기존 환경변수 수정이 용이함
  - setenv
  - prepend-path
  - append-path
  - set-alias
- 환경변수 등록 및 삭제가 용이함
  - module load <packages>
  - module rm <packages>
  - module purge
- 동일한 package의 버전간 전환이 용이함
  - module switch <package>

Environment module을 활용사례

- OpenFOAM 사용환경을 위한 module file 작성
  - 만드는 방법
    - printenv 명령으로 OpenFOAM 환경 설정 전/후 내용 비교
    - OpenFOAM에만 사용되는 환경변수 추출 (약 100개)
      - Vim 이용하면 간단하게 정리됨
    - Modules 파일 문법에 맞게 정리

OpenFOAM 환경변수 목록 작성 script

```
$ printenv | sort > env0
$ source $WM_PROJECT_DIR/etc/bashrc
$ printenv | sort > env1
$ diff -urN env0 env1 | sed -e '/^[^+]|+/ d' -e '/^++/d' -e 's/^+//' | sort
```

Environment module 예

OpenFOAM-2.3.0 module file 내용 일부

```
proc ModulesHelp { } {
   global version foamRoot

   puts stderr "   This module defines environment variables, aliases and add PATH, LD_LIBRARY_PATH for OpenFOAM"
   puts stderr "   Version $version   "
}
prereq   intel mvapich2
conflict OpenFOAM FOAM-extend
module-whatis   "OpenFOAM 2.3.0"

# for Tcl script use only
set             version                 2.3.0
set             foamRoot                /opt/OpenFOAM

# begin modules
setenv WM_PROJECT               OpenFOAM
setenv WM_PROJECT_VERSION       $version
setenv WM_COMPILER              $env(COMPILER_OF_TYPE)
…
setenv PINC                     [exec mpicc -show -cc= -nativelinking]
setenv PLIBS                    [exec mpicc -show -cc= | sed "s%$env(PINC)%%"]
…
prepend-path PATH               $env(WM_THIRD_PARTY_DIR)/platforms/$env(WM_ARCH)$env(WM_COMPILER)/gperftools-svn/bin
prepend-path PATH               $env(FOAM_USER_APPBIN)
…
prepend-path LD_LIBRARY_PATH    $env(FOAM_LIBBIN)/dummy
prepend-path LD_LIBRARY_PATH    $env(FOAM_EXT_LIBBIN)
…
set-alias    app                "cd $env(FOAM_APP)"
set-alias    foam               "cd $env(WM_PROJECT_DIR)"
…
```

Job 제출용 (LSF 기준) script

of_sub.py

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

# job submitting python script inspired by perl script of_sub
# Author : SeongMo Yeon
# Date   : 07/28/2015

import sys
import os
from subprocess import Popen, PIPE
from optparse   import OptionParser
import re

if __name__ == '__main__':

 usage = "usage: %prog [options] <solver>"
  parser = OptionParser(usage=usage)
  parser.add_option("-n", "--np", type="string", dest="ncpu", default="64", help="Number of cpu cores for parallel computation. default cores is 64.")
  parser.add_option("-v", "--version", type="string", dest="version", default="of4x", help="OpenFoam version. default version is OpenFOAM 4.x")
  parser.add_option("-V", "--listversion", action="store_true", dest="listVersion", help="List of openfoam versions available")

  (options, args) = parser.parse_args()

  foamVersion={'of230':'OpenFOAM/2.3.0'}

  if(options.listVersion):
    print('*** Available OpenFOAM Versions ***')
    print('-----------------------------------------')
    print('  '+'{0:<15s}'.format('Keyword') + '  |  ' + '{0:<15s}'.format('Module Name'))
    print('-----------------------------------------')
    for k, v in foamVersion.items():
        if (options.version == k):
            print('* '+'{0:<15s}'.format(k) + '  |  ' + '{0:<15s}'.format(v))
        else:
            print('  '+'{0:<15s}'.format(k) + '  |  ' + '{0:<15s}'.format(v))
    print('-----------------------------------------')
    exit(0)

command='bsub'
```

Job 제출용 script

continue

```
if(len(args)==0):
    print('Please specify solver')
    exit(0)
  else:
    solver = ''
    for arg in args:
        solver = solver  + ' ' + arg

  if (int(options.ncpu) == 1):
    solver = solver
  else:
    solver = solver + ' ' + '-parallel' # it is OpenFoam solver's parallel option

  jobname=os.path.basename(os.getcwd())

  template=('#!/bin/bash\n'
            '#BSUB -q openfoam\n'
            '#BSUB -J $jobname\n'
            '#BSUB -Jd description\n'
            '#BSUB -P Project\n'
            '#BSUB -n $ncpu\n'
            '#BSUB -o %J.lsflog\n'
            'module purge\n'
            'module load use.own\n'
            'module load gcc/4.9.3\n'
            'module load openmpi/1.10.3\n'
            'module load $foamVersion\n'
            'mpirun -np $ncpu -hostfile $LSB_DJOB_HOSTFILE $solver')

  subfile=template.replace('$jobname', jobname)
  subfile=subfile.replace('$ncpu', options.ncpu)
  subfile=subfile.replace('$foamVersion', foamVersion[options.version])
  subfile=subfile.replace('$solver', solver)

  print(subfile)

  p = Popen(command,stdin=PIPE, stdout=PIPE, stderr=PIPE, shell=False)
  out, err = p.communicate(subfile)
  print("")
  print("------------------- Job Submission Results -------------------")
  print("** BSUB stdout: " + out.replace('\n','')) # remove the redundant last newline character
  print("** BSUB stderr: " + err.replace('\n','')) # remove the redundant last newline character
  print("--------------------------------------------------------------")
```

요약 및 결론

- Environment modules를 이용한 동적 환경변수 관리
- 시스템 관리 측면
  - 일반유저에게 동일한 사용환경을 제공가능
  - package 추가 및 관리가 용이
  - OpenFOAM 버전 관리가 용이해짐
- 일반 유저 측면
  - 지저분한 PATH, LD_LIBRARY_PATH 관리가 필요없음
  - .bashrc 관리가 용이해짐
  - 환경변수 오염이 없어 shell 변경없이 OpenFOAM 실행이 가능해짐

OpenFOAM 개발을 위한 IDE 환경 구축

C++ 특징

- OOP
  - Encapsulation
  - **Inheritance**
  - Polymorphism
  - Abstraction
- Generic programming
  - **Template**

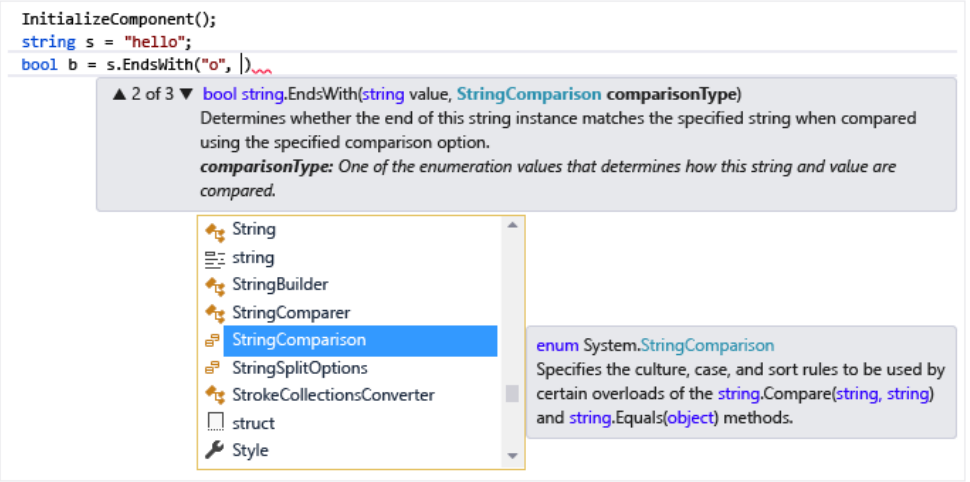
Issues

- 복잡한 계층관계
- Member func., var. 에 대한 정보파악이 어려움
- Template
  - Cryptic Info.

Needs

- Source navigation
  - Def., decl. 간의 전환
  - 소스코드간 이동
- List of members
- Easy Template handling

Integrated Development Environment (IDE) / Intellisense



OS 별 대표적인 C++ 개발환경

MS Windows

Visual Studio

OS X

Xcode

Linux

Linux is IDE itself

- Keep It Simple and Short (KISS) 철학
- Text editor + tagging tools (ctags, cscope, global)

Gcc

?

Why?

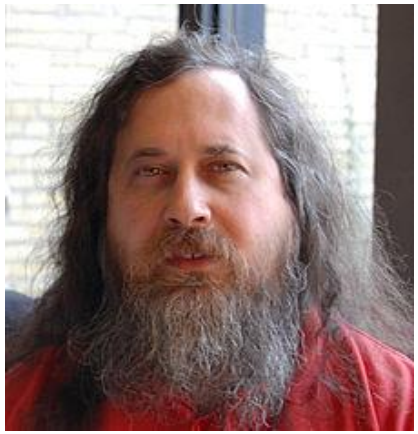
기존 tagging tool (indexer)의 한계

- Ctags, cscope, global (gtags)
- Pattern matching 기반 자동완성 제공
  - Rough guess and many candidates irrelevant
- Project 설정을 반영못함
  - Project 정보: Make, Cmake에 이미 있음
    - Compile flag
    - Define문을 통한 conditional compilation
- Template에는 사용불가
  - Compile time type check

- Context 기반 tagging tool 필요
  - Compiler's Abstract Syntax Tree (AST)에의 접근 필요

Gcc에서 안됨
Why?

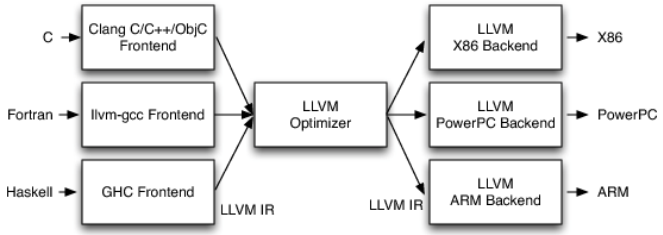
GCC AST 접근 차단의 배경



- **Richard M. Stallman (aka RMS)**
  - Free Software Foundation (FSF) 설립자 및 Copyleft 개념 창시자
    - GPL license (or virus)
      - Recursive license
    - GNU gcc/gdb 저자
    - Emacs 저자이자 Emacs교 교주 (Vim 에디터 종교전쟁)
  - Free software vs. Open source
    - Debate between RMS vs. Eric S. Raymond
      - Protection S/W right from proprietary use (Free software)
        - » Crashing my car so nobody wants to steal it
      - Better code to all anyway (Open source)
    - Parsing 정보 (Abstract Syntax Tree)에 대한 API 부재
      - **Gcc를 이용한 완벽한 intellisense 구현 불가능**
  - Gcc의 Intellisense 기능 지원 문제로 현재까지 격론 중 (https://lwn.net/Articles/629259/)

Clang 등장

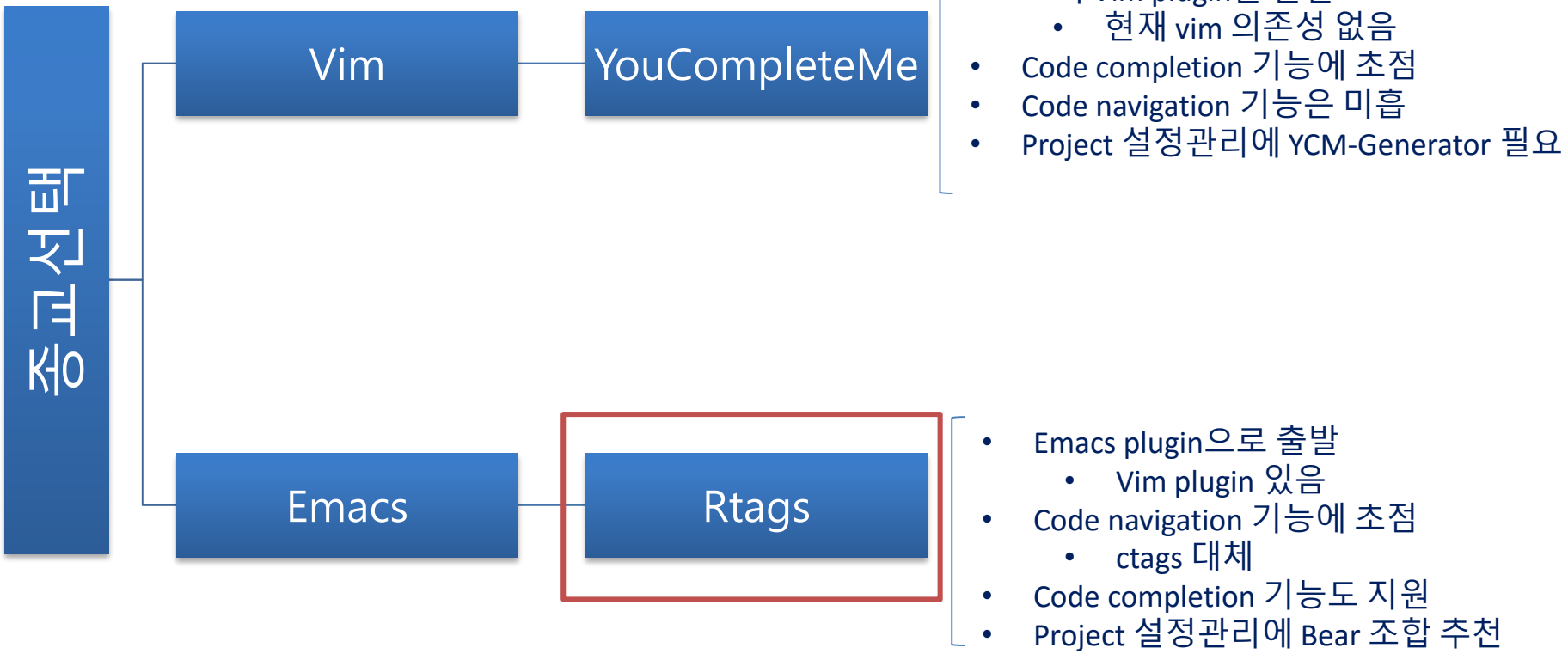
- Clang 이란
  - LLVM project
    - U of Illinois, Chris Lattner 석사 논문에서 시작
    - Compiler를 단계별로 모듈화시켜 컴파일러 작성, 최적화 과정 분리
      - Compiler(front-end)는 LLVM IR 로의 변환과정만 처리
      - 통일된 byte code (LLVM IR)로 부터 동일한 최적화 과정 적용가능
      - Multiplatform compiler 작성이 용이
  - LLVM 기반 C/C++ compiler
  - GPL license 제약으로 부터 벗어나기 위해 BSD license 채용
    - FreeBSD에서 gcc대신 clang 기반으로 system 구성시작
  - Sponsored by Apple
    - OS X(BSD 계열)의 gcc를 대체할 compiler 찾고 있었음
    - **Xcode**



- Intellisense에의 활용
  - Libclang에서 제공하는 **C++ parsing 결과 (AST)**에 접근할 수 있는 API 제공
  - Intellisense 기능 (indexer)을 구현하는 다양한 프로젝트 등장
    - **YouCompleteMe**
    - **Rtags**
    - Clang complete
    - Autocomplete-clang
    - **Xcode (상용)**

LLVM의 대표적인 성공작

고려사항
- Terminal 기반 작업가능할 것
  - No GUI, No XWindow
- Lightweight S/W 일것
  - Eclipse 등의 IDE 불필요



- 초기 Vim plugin을 출발
  - 현재 vim 의존성 없음
- Code completion 기능에 초점
- Code navigation 기능은 미흡
- Project 설정관리에 YCM-Generator 필요

- Emacs plugin으로 출발
  - Vim plugin 있음
- Code navigation 기능에 초점
  - ctags 대체
- Code completion 기능도 지원
- Project 설정관리에 Bear 조합 추천

Plugin 설치 절차

- LLVM + Clang 설치
- VIM
  - Plugin 관리도구 Vundle 설치 (https://github.com/VundleVim/Vundle.vim.git)
  - YouCompleteMe 설치 (http://valloric.github.io/YouCompleteMe 참고)
  - YCM-Generator 설치 (https://github.com/rdnetto/YCM-Generator)
    - 현재 OpenFOAM 지원 (by 연성모)
  - .vimrc 설정
- Emacs
  - Rtags 설치 (https://github.com/Andersbakken/rtags)
  - Rtags 및 code complete도구 package 설치
    - Rtags
    - Auto-complete
    - Yasnippet
    - Flycheck
    - Company
    - Company-quickhelp
  - Bear (Build EAR) 설치 (https://github.com/rizsotto/Bear)
    - Gcc, clang + intel (by 연성모) compiler 지원
  - .emacs 설정

설정 파일 예

.vimrc

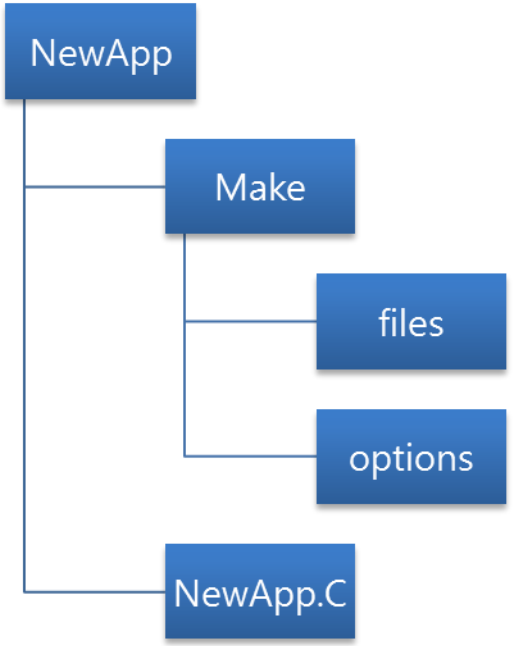
```
set nocompatible
filetype off
set rtp+=~/.vim/bundle/Vundle.vim

call vundle#begin()

Plugin 'VundleVim/Vundle.vim'
Plugin 'Valloric/YouCompleteMe'

let g:ycm_confirm_extra_conf = 0
"let g:ycm_server_keep_logfiles = 1
"let g:ycm_server_log_level = 'debug'
let g:ycm_key_list_select_completion = ['<C-j>', '<Down>']
let g:ycm_key_list_previous_completion = ['<C-k>', '<Up>']
let g:ycm_autoclose_preview_window_after_completion = 1

nnoremap <leader>g :YcmCompleter GoTo<CR>
nnoremap <leader>gg :YcmCompleter GoToImprecise<CR>
nnoremap <leader>d :YcmCompleter GoToDeclaration<CR>
nnoremap <leader>t :YcmCompleter GetType<CR>
nnoremap <leader>p :YcmCompleter GetParent<CR>

call vundle#end()

filetype plugin indent on
hi Comment cterm=bold ctermfg=blue
syntax on
"set number
set encoding=cp949
set fileencodings=utf-8,cp949
set langmenu=cp949
set expandtab
set tabstop=4
set shiftwidth=4
set nobackup

set t_Co=25
```

.emacs

```
(when (require 'rtags nil :noerror)
  ;; make sure you have company-mode installed
  (require 'company)
  (require 'company-rtags)
  (add-hook 'c-mode-common-hook 'rtags-start-process-unless-running)
  (add-hook 'c++-mode-common-hook 'rtags-start-process-unless-running)

  ;; install standard rtags keybindings. Do M-. on the symbol below to
  ;; jump to definition and see the keybindings.
  (rtags-enable-standard-keybindings)
  ;; comment this out if you don't have or don't use helm
  (setq rtags-use-helm t)
  ;; company completion setup
  (setq rtags-autostart-diagnostics t)
  (rtags-diagnostics)
  (setq rtags-completions-enabled t)
  ;; (push 'company-rtags company-backends)
  (add-to-list 'company-backends 'company-rtags)
  ;; use rtags flycheck mode -- clang warnings shown inline
  (require 'flycheck-rtags)
  ;; c-mode-common-hook is also called by c++-mode
  (add-hook 'c-mode-common-hook 'setup-flycheck-rtags)
  (add-hook 'c++-mode-common-hook 'setup-flycheck-rtags)
  )
(when (require 'rtags nil 'noerror)
  (add-hook 'c-mode-common-hook
            (lambda ()
              (when (rtags-is-indexed)
                (local-set-key (kbd "M-;") 'rtags-find-symbol-at-point)
                (local-set-key (kbd "M-:") 'rtags-find-symbol)
                (local-set-key (kbd "M-'") 'rtags-find-references-at-point)
                (local-set-key (kbd "M-\"") 'rtags-find-references)
                (local-set-key (kbd "M-,") 'rtags-location-stack-back)
                (local-set-key (kbd "<C-tab>") (function company-complete))
                ))))
```

Vim

```
$cd NewApp
$config_gen –e –f .
$vim NewApp.C
```



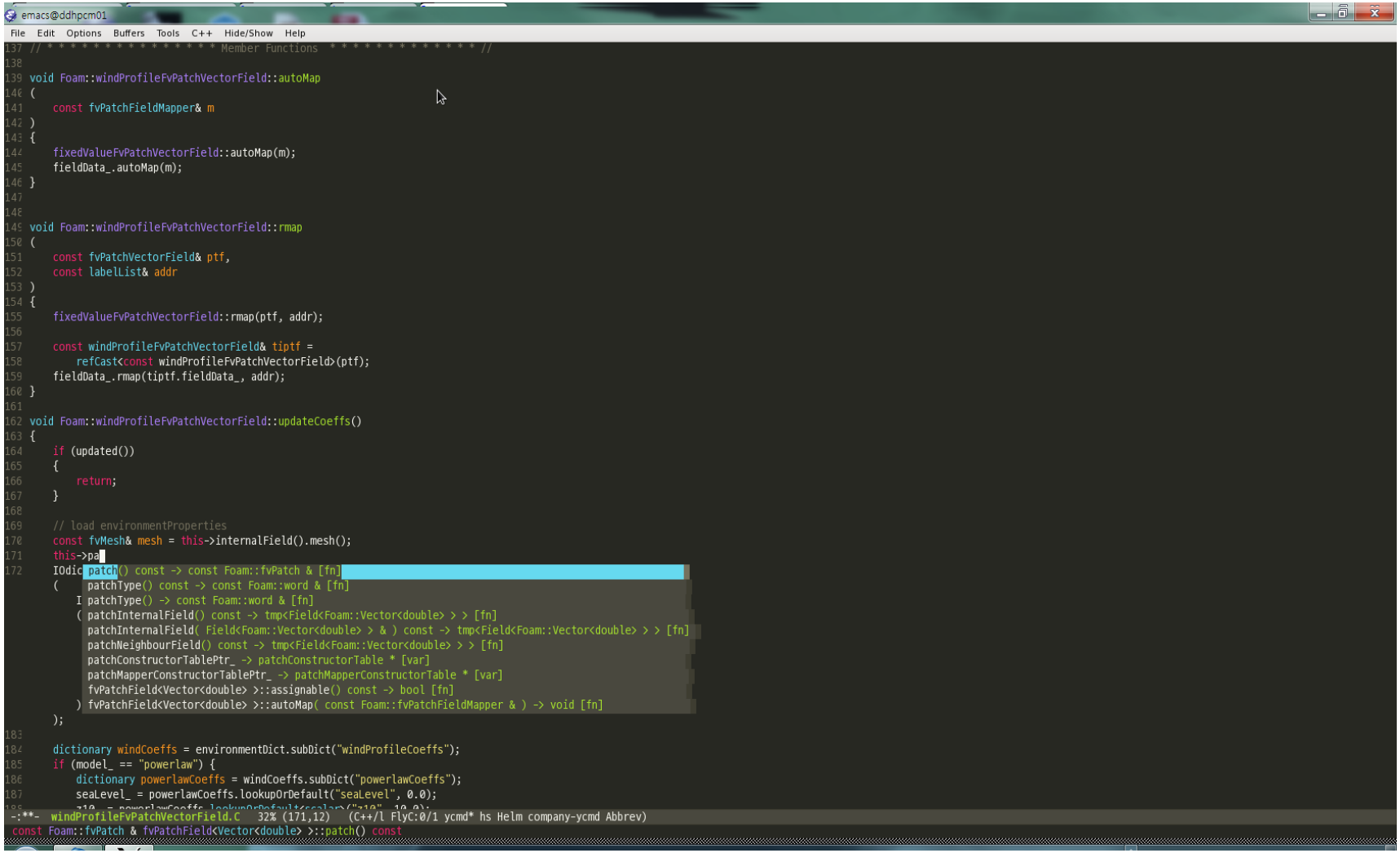
Emacs

```
$cd NewApp
$rdm --daemon
$bear wmake
$rc –J
$emacs NewApp.C
```

System service로 등록가능하므로 로그인시에만 실행되도록 할 수 있음

VIM/Emacs 실제 코드 작성 예

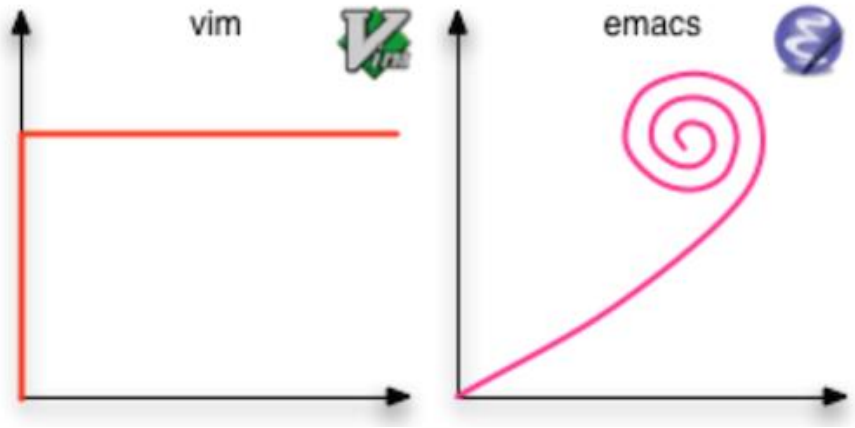
YouCompleteMe

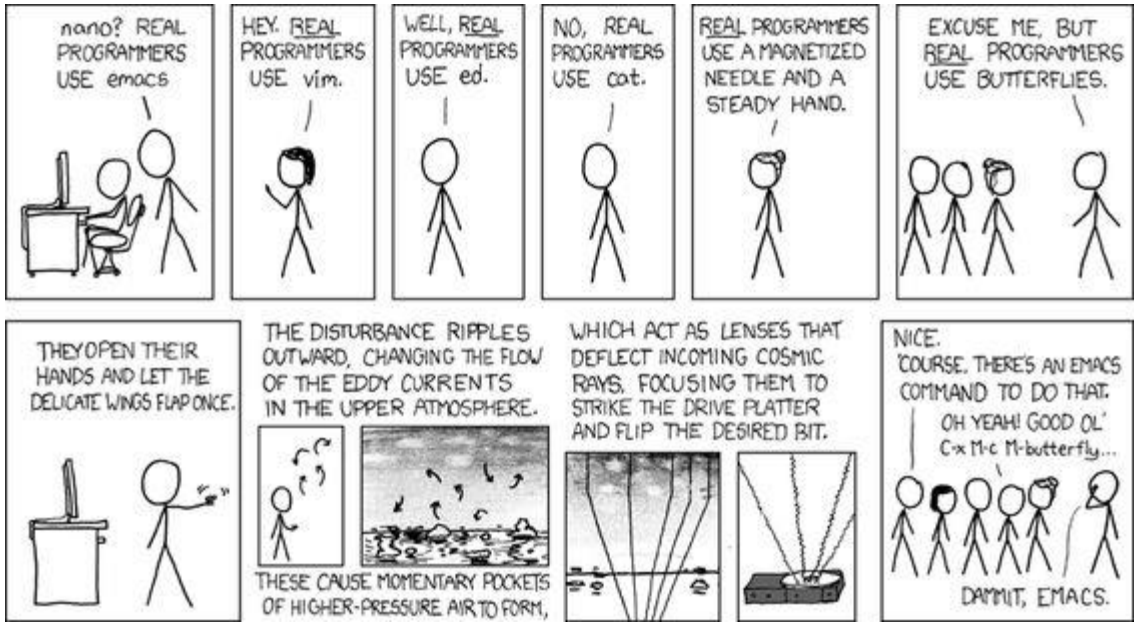


Rtags

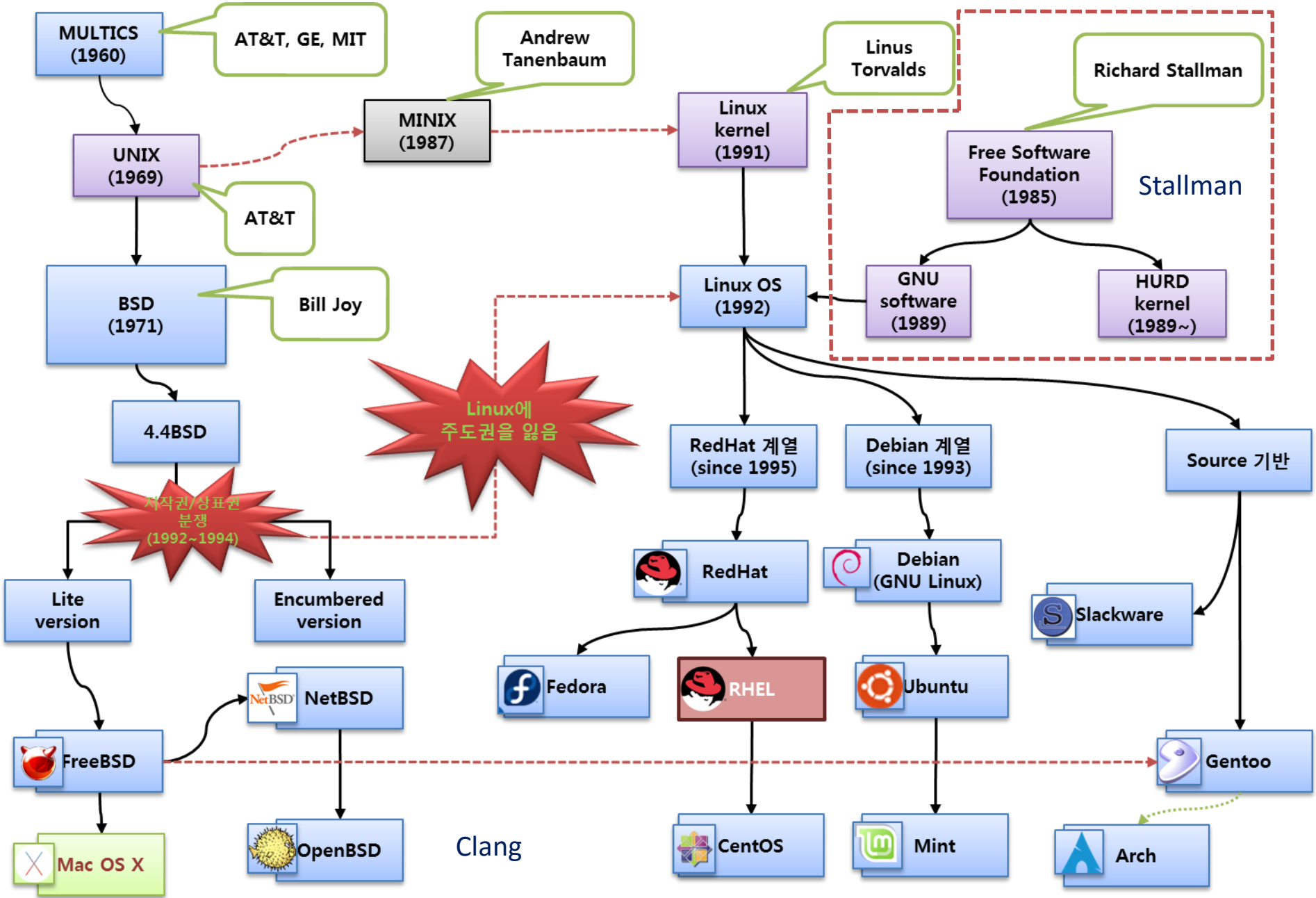
요약 및 결론

- Intellisense
  - 코드 자동완성 기능
  - 생산성 향상에 유리
  - 코드 분석에 유리
  - OpenFOAM 개발 진입 장벽을 낮춤
- Clang compiler 등장
  - Linux 에서 context 기반 tagging system 구현가능
  - 적당한 흉내가 아닌 본격적인 code assistant 기능 구현 가능
  - YouCompleteMe로 가벼운 vim IDE 구현이 가능
  - Rtags 사용으로 거의 완벽한 emacs IDE 구현이 가능
- 문제점
  - Vim, emacs를 배워야함




nano? REAL PROGRAMMERS USE emacs
HEY. REAL PROGRAMMERS USE vim.
WELL, REAL PROGRAMMERS USE ed.
NO, REAL PROGRAMMERS USE cat.
REAL PROGRAMMERS USE A MAGNETIZED NEEDLE AND A STEADY HAND.
EXCUSE ME, BUT REAL PROGRAMMERS USE BUTTERFLIES.
THEY OPEN THEIR HANDS AND LET THE DELICATE WINGS FLAP ONCE.
THE DISTURBANCE RIPPLES OUTWARD, CHANGING THE FLOW OF THE EDDY CURRENTS IN THE UPPER ATMOSPHERE.
THESE CAUSE MOMENTARY POCKETS OF HIGHER-PRESSURE AIR TO FORM,
WHICH ACT AS LENSES THAT DEFLECT INCOMING COSMIC RAYS, FOCUSING THEM TO STRIKE THE DRIVE PLATTER AND FLIP THE DESIRED BIT.
NICE.
'COURSE. THERE'S AN EMACS COMMAND TO DO THAT.
OH YEAH! GOOD OL' C-x M-c M-butterfly...
DAMMIT, EMACS.

Linux 계보


MULTICS (1960)
AT&T, GE, MIT
Andrew Tanenbaum
MINIX (1987)
Linus Torvalds
Linux kernel (1991)
Richard Stallman
Free Software Foundation (1985)
Stallman
UNIX (1969)
AT&T
BSD (1971)
Bill Joy
Linux OS (1992)
GNU software (1989)
HURD kernel (1989~)
Linux에 주도권을 잃음
4.4BSD
저작권/상표권 분쟁 (1992~1994)
RedHat 계열 (since 1995)
Debian 계열 (since 1993)
Source 기반
Lite version
Encumbered version
RedHat
Debian (GNU Linux)
Slackware
Fedora
RHEL
Ubuntu
NetBSD
FreeBSD
Gentoo
Mac OS X
OpenBSD
Clang
CentOS
Mint
Arch