Facultad de Ingeniería en Electricidad y Computación

Programación de Sistemas CCPG1051

Federico Domínguez, PhD.

Unidad 2 – Sesión 2: Herramientas de gestión de desarrollo

Agenda

Editor de texto: vi

Herramienta de gestión compilación: make

vi: Editor POSIX de sistemas UNIX/LINUX

vi es un editor de texto incluido por defecto en casi todas las distribuciones de Linux.

- Siempre disponible
- Rápido y de bajos recursos
- Muy popular en la comunidad código abierto

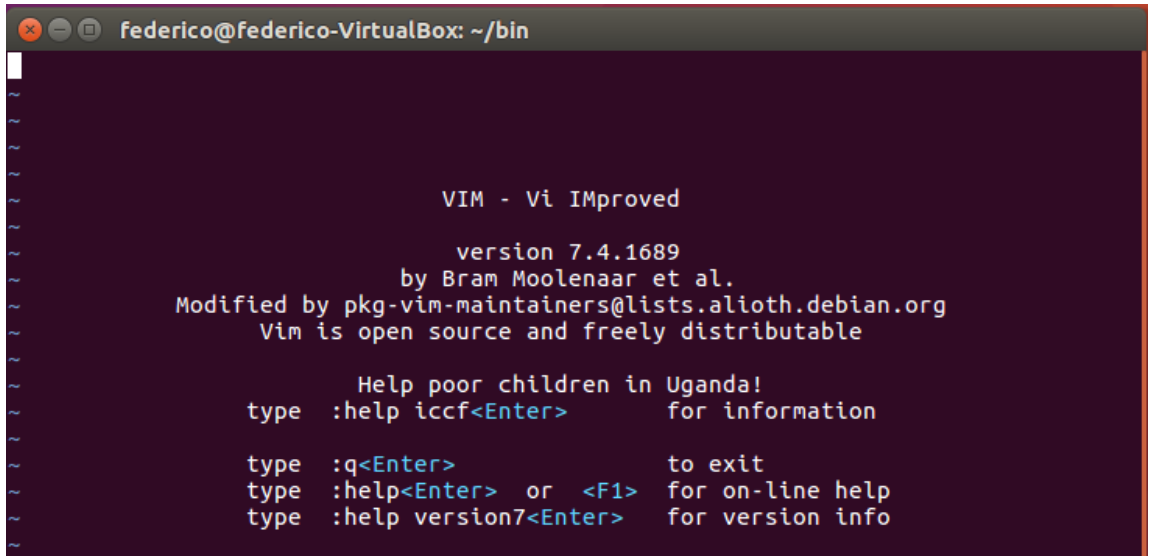
Otra opción: nano

vim: versión mejorada de vi

Incluida en la mayoría de distribuciones

Usualmente, vi es un alias a una versión de vim con capacidades mínimas, recomendado instalar vim:

- Ubuntu: `sudo apt install vim`
- Centos/Fedora: `yum install vim`



vi tiene varios modos de trabajo

Empieza en modo de comando

Modo de inserción (a,A,i,o,O)

Modo de ex command (:)

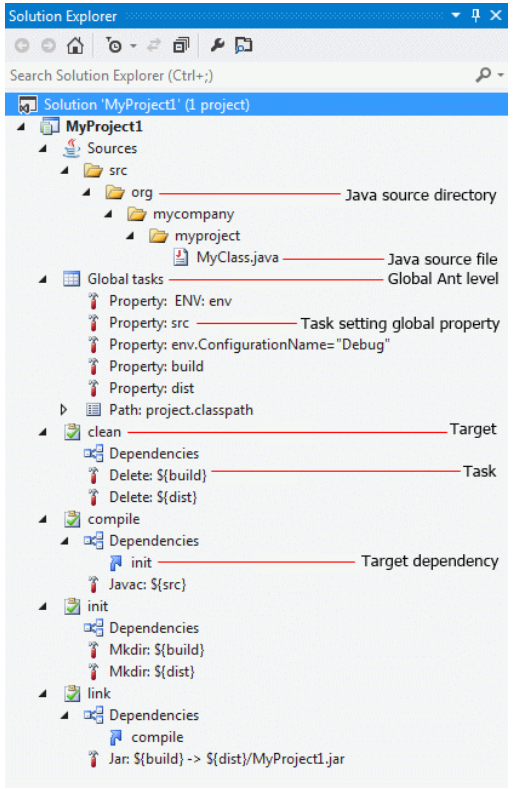
a	Enter insert mode. Characters typed are inserted into the file just after the current character.
A	Enter insert mode. Characters typed are inserted into the file at the end of the current line.
dd	Delete the current line.
h	Same as left arrow key.
i	Enter insert mode. Characters typed are inserted into the file just before the current character.
j	Same as down arrow key.
k	Same as up arrow key.
l	Same as right arrow key.
o	Open a new line beneath the current line, and enter insert mode.
O	Open a new line above the current line, and enter insert mode.
p	Put back most recently deleted text.
r	Replace the current character with the next character typed.
u	Undo previous change.
x	Delete the current character.
ctl-B	Backward screen.
ctl-D	Forward half-screen.
ctl-F	Forward screen.
ctl-U	Backward half-screen.
:q!	Exit without writing the file.
:wq	Exit, writing the file.
:w	Write the file, but don't exit.
.	Repeat previous command.
/	Search for a string. Type the string to search for after typing /, followed by carriage return. / immediately followed by return repeats the previous search.
?	Same as /, but search backward.
esc	The escape character is used to exit insert mode. Be sure to exit insert mode before trying to issue commands.

Herramientas de gestión de compilación

Conocidos como *build tools*, son necesarios cuando los proyectos de desarrollo de software escalan en complejidad y tamaño.

Build tools típicamente automatizan:

- Compilación de código fuente
- Empaquetamiento
- Pruebas y despliegue



GNU Make es la herramienta de gestión de compilación *de facto* en Linux.

GNU Make nació por la necesidad, de programadores en el proyecto GNU, de actualizar rápidamente todos los binarios ejecutables una vez que se modificaba el código fuente.

Make esta diseñado para actualizar un archivo automáticamente siempre y cuando otros en los que depende sean modificados.

Make utiliza archivos de configuración llamados *Makefiles* para guiar la compilación de un proyecto.

Make es parte de una cadena de herramientas conocida como el **GNU Build System** (*Autotools*).

- *Autoconf* es también parte de *Autotools* y es capaz de generar archivos *Makefile*.
- El objetivo de *Autotools* es gestionar la portabilidad de código fuente entre plataformas.

La compilación de estos tres archivos puede ser automatizada con *make*.

hellomake.c

```
#include <hellomake.h>

int main() {
  // call a function in another file
  myPrintHelloMake();

  return(0);
}
```

hellofunc.c

```
#include <stdio.h>
#include <hellomake.h>

void myPrintHelloMake(void) {

  printf("Hello makefiles!\n");

  return;
}
```

hellomake.h

```
/*
example include file
*/

void myPrintHelloMake(void);
```

```
gcc -o hellomake hellomake.c hellofunc.c -I.
```

Al ejecutar *make*, el programa busca un archivo con el nombre *Makefile* en el directorio local.

Archivo Makefile:

- Contiene una lista de reglas

```
target [target ...]: [component ...]
Tab [command 1]
        .
        .
        .
Tab [command n]
```

Ejemplo:

```
hellomake: hellomake.c hellofunc.c
        gcc -o hellomake hellomake.c hellofunc.c -I.
```

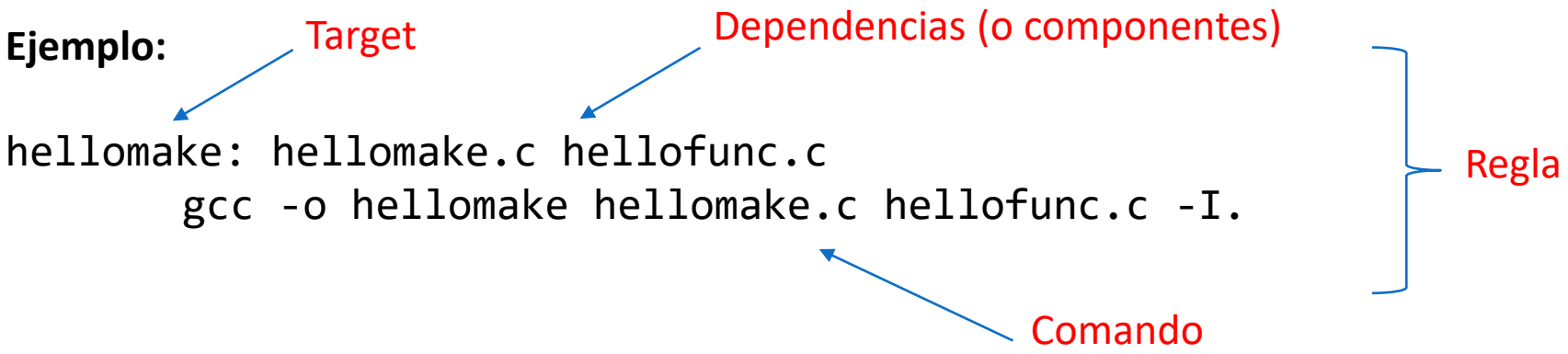
Al ejecutar *make*, el programa busca un archivo con el nombre *Makefile* en el directorio local.

Archivo Makefile:

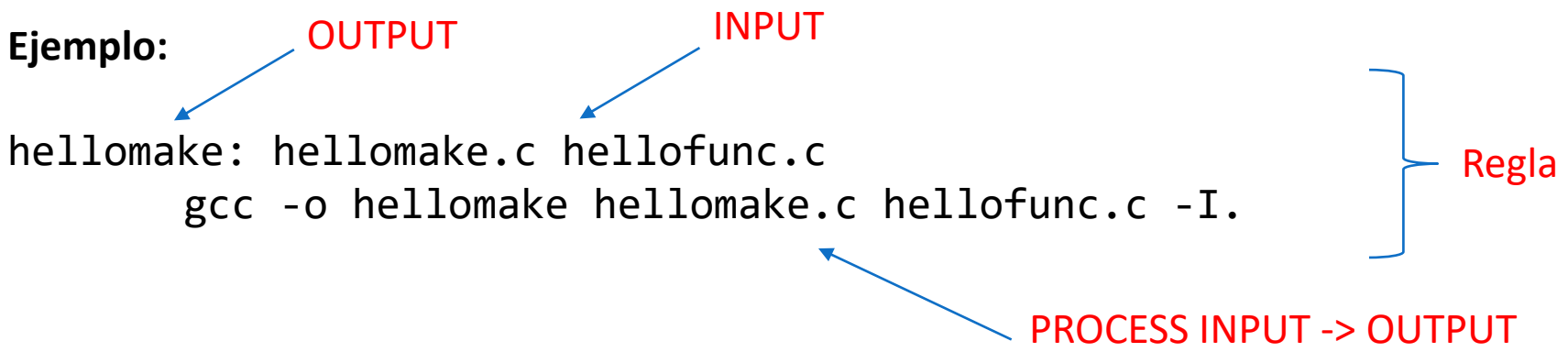
- Contiene una lista de reglas

```
target [target ...]: [component ...]
Tab [command 1]
        .
        .
        .
Tab [command n]
```

Ejemplo:



Una **regla** especifica un archivo (**target**) que debe ser creado usando un comando o comandos (**command**) y una dependencia o dependencias (**component**).



Demostración

Referencias uso de *make*

Documentación oficial: https://www.gnu.org/software/make/

Tutorial básico: http://www.cs.colby.edu/maxwell/courses/tutorials/maketutor/