Make

Egilea: Kepa Bengoetxea
Kepa.bengoetxea@ehu.es

Make

- **Proiektu handietan** erabiltzen da, hainbat fitxero iturri(.c) eta modulu(.o) dauden proiektuetan. “.o” moduluen arteko menpekotasunak kontutan hartzen ditu.
- **Make komandoak** fitxategi multzo bat konpilatu behar den zehazten du, fitxategien data kontutan hartuz. Fitxategi helburuaren data (exekutagarria, edo bitarteko moduluren bat adibidez), iturri fitxategien data baino zaharragoa izanez gero, komando multzo bat exekutatuko da. Komando hauen helburua, helburu fitxategia eguneratzea izango da.

Motibazioa

```
$vi define.h
#ifndef _DEFINE_H
#define _DEFINE_H
#define PI 3.1415
#endif
```

```
$vi bolumena.h
#ifndef _BOLUMENA_H
#define _BOLUMENA_H
int kubo (int x);
float esfera (int x);
#endif
```

```
$vi azalera.h
#ifndef _AZALERA_H
#define _AZALERA_H
int karratu (int x);
float zirkulu (int x);
#endif
```

Motibazioa

```
$vi  azalera.c
#include "define.h"
int karratu(int x)
{return x*x;}
float zirkulu(int x)
{return PI*x*x;}
```

```
$vi bolumena.c
#include "define.h"
int kubo (int x)
{return x*x*x;}
float esfera (int x)
{return 4*PI*x*x*x/3;}
```

Motibazioa

```
$vi geometria.c

#include <stdio.h>

#include "azalera.h"

#include “bolumena.h"

int main(){

int aldea,erradio;

printf(“Alde bat sartu \n");

scanf("%d",&aldea);
```

```
printf(“erradio bat sartu\n");

scanf("%d",&erradio);

printf(“Kuboa %d  da \n",kubo(aldea));

printf(" Esfera %f da \n",esfera(erradio));

printf(“Karratua %d da
\n“,karratua(aldea));

printf(“zirkulu %f  da \n",zirkulu(erradio));

return 0;

}
```

Motibazioa

```
gcc -I. -c bolumena.c -o bolumena.o
gcc -I. -c azalera.c -o azalera.o
gcc -I. -c geometria.c -o geometria.o
gcc bolumena.o azalera.o geometria.o -o geometria
```

Motibazioa

$make (makefile fitxategian dagoan gidoia interpretatu)

makefile gidoiaren adibide bat:

```
geometria:azalera.o bolumena.o geometria.o
    gcc geometria.o azalera.o bolumena.o -o geometria
bolumena.o: bolumena.c
    gcc -I. -c bolumena.c -o bolumena.o
azalera.o: azalera.c
    gcc -I. -c azalera.c -o azalera.o
geometria.o: geometria.c
    gcc -I. -c geometria.c -o geometria.o
```

Motibazioa

Iturriak konpilatu eta instalatzeko metodo unibertsala. Linux edozein banaketarako balio du.

1.-Deskargatu, paketea deskonprimitu. Adibidez:tar -zxvf paketea.tar.gz
2.-Irakurri:README,INSTALL
3.-./configure //scriptak makefile bat sortzen du
4.-**make** //konpilatu eta batzutan instalatu
5.-**make install** (aukerakoa)
6.-**make clean** (aukerakoa)
Adibidez:How to compile Reaver under Ubuntu 12.04 (and aircrack-ng)http://securit.se/en/2012/03/kompilera-reaver-ubuntu-12-04/

Motibazioa

Iturri kodea:

sistema-aplikazioa optimizatu egin ahal da konpilatzerakoan

Adibidez: AMD K6-2, K6-3, Athlon eta Duron mikroprozesadoreak dituzuenok eta 3DNow instrukzioen abantailak Xmms aplikazioan erabili nahi izanez gero, aukera hauekin konpilatu beharko dituzue:

./configure -prefix=/usr/local -enable-3dnow

Kode Bitarra:

Konpilatuta dauden aplikazioak

Normalean bitarrak liburutegi dinamikoak erabiltzen ditu. ldd-k bitar batek zein liburutegi behar dituen informatzen gaitu. Adibidez: $/usr/bin/ldd /usr/bin/gedit

Makefile(Guk egindakoa)

$make (makefile fitxategian dagoan gidoia interpretatu)

makefile gidoiaren adibide bat:

```
geometria:azalera.o bolumena.o geometria.o
    gcc geometria.o azalera.o bolumena.o -o geometria
bolumena.o: bolumena.c
    gcc -I. -c bolumena.c -o bolumena.o
azalera.o: azalera.c
    gcc -I. -c azalera.c -o azalera.o
geometria.o: geometria.c
    gcc -I. -c geometria.c -o geometria.o
```

Makefile(Egindako bat)

```
INC-DIR = -I. -I../include -I/usr/X11R6/include
    -I/usr/local/include
LIB-DIR = -L. -L/usr/X11/lib -L -L/usr/local/lib -L/usr/X11R6/lib
GCC = gcc
CFLAG = -ansi -pedantic -Wall
all: eem
eem: eem.o feem.o
        $(GCC) $(CFLAG) $(LIB-DIR) -o eem eem.o feem.o -lRai
    -lforms -lX11 -lm
eem.o: eem.c
        $(GCC) $(CFLAG) $(INC-DIR) -c eem.c
feem.o: feem.c
        $(GCC) $(CFLAG) $(INC-DIR) -c feem.c
clean:
        rm -f *.o
```

Make

- Helburu fitxategia “helburua” da eta iturburu fitxategiak “aurrebaldintzak”. Aurrebaldintzetako bat helburu fitxategia baino berriagoa baldin bada (edo helburua ez badago) komandoak exekutatuko dira. Aurrebaldintza guztiak helburu fitxategia baino zaharragoak izanez gero, orduan komandoak ez dira exekutatuko eta helburu fitxategia eguneratua bezala, hartuko da.
- Oraingo lan katalogoan, fitxategi bat sortu beharko genuke **Makefile, GNUmakefile, makefile** izenarekin. Fitxategi honek , make komandoak bere lana behar bezala egiteko beharrezkoa duen informazioa du. **Makefile**-a dugunean, egin behar den gauza bakarra: ‘make ‘ tekleatu eta helburu fitxategi berri bat sortzeko beharrezko diren komandoak automatikoki exekutatuko dira.

Make

$make (helburu nagusia bilatzen du)

helburu nagusia, “:” duen lehenengo arauan gelditu eta “:”ren ezkerrean dagoan hitza izango da. Adb:”:”z aurkitutako lehenengo araua ”all: eem” denez, helburua “all” izango da.

make, helburu nagusia bilatzen du. Adb: all: eem eta “eem” menpekotasuna duela aurkitzen du. Orain “eem” helburu bezala dagoan lerro bat bilatzen du.”eem” helburu bezala existituz gero, “eem” helburua prozesatzen hasiko da. Eta aurreko helburua “all” menpekotasun guztiak prozesatu arte utziko da. Menpekotasunen bat, helburu bezala aurkitzen ez badu, orduan prozesatzen ari den katalogo beran fitxategi bezala bilatuko du. Fitxategia ez bada agertzen, akatsa emango du eta agertuz gero fitxategiaren data helburuaren data baino berriagoa izanez gero dagokion ordena exekutatuko da eta hierarkian gorantza egingo du helburu nagusira arte. Menpekotasunik ez duten helburuak egon daitezke, adibidez: clean.

Make

- Abantailak:
  - Make-ekin konpilatzerakoan, ikutu ditugun fitxategiaren menpe dauden fitxategiak bakarrik birkonpilatuko dira.
  - Konpilatzeko erabiltzen diren komandoak bere parametro guztiekin gordetzen ditu, liburutegiak , goiburu fitxategiak (.h), etab aurkitzeko.
  - Ez ditugu idatzi behar izango buruz jakin behar ditugun aukerekin eginiko konpilaziorako lerro luzerik, edo behintzat behin bakarrik idatzi beharko dugu makefile fitxategia.

Make

- Abantailak:
  - 'Make'ek iturrietariko batek exekutagarriak baino data berriagoa badu soilik konpilatzen du.
  - Gainera 'make'ek luzapenak eta konpilatzaileak ulertzen ditu. Adb: make KaixoMundua, KaixoMundua.c fitxategia gcc konpilatzaileaz konpilatukoa da. Orain luzapena aldatu eta KaixoMundua.cpp deituz gero, make egiterakoan, c++ ko konpilatzaileaz (g++) konpilatuko da.

```
$ls ->KaixoMundua.c
$make KaixoMundua
$./KaixoMundua
```

```
less makefile
KaixoMundua: KaixoMundua.o
        gcc   KaixoMundua.o  -o
              KaixoMundua
KaixoMundua.o: KaixoMundua.c
        gcc -c -o KaixoMundua.o
              KaixoMundua.c
```

Make

'Make'ek Makefile fitxategia irakurtzen du, eta

honek arau batzuk ditu:

- Iturri programa bat baino gehiago ditugunean, zer egitea nahi dugun esan behar diogu:

  Helburua: menpekotasuna1 menpekotasuna2...

  <tab> komandoa1

  <tab> komandoa2

  <tab>...

Make

Helburua: Exekutagarri edo liburutegiaren izena.

Menpekotasuna<i> gure helburuaren aurretik egin behar den beste helburu baten izena edota gure helburuak beharrezko dituen menpekotasun fitxategiak dira.

<tab> tabuladorea da.

Komando<i> da exekutatu behar duguna gure helburua sortzeko.

Make

/home/euiti/include/funtzioa1.h
```
void idatziKaixoMundua();
```

/home/euiti/iturri/funtzioa1.c
```
#include <stdio.h>
void idatziKaixoMundua()
{printf (“Kaixo Munduari/n”);}
```

/home/euiti/iturri/KaixoMundua.c
```
#include “/home/euiti/include/funtzioa1.h”
int main(void)
{idatziKaixoMundua();
 return0}
```

Make

/home/euiti/iturri gaudela suposatuko dugu

KaixoMundua: KaixoMudua.c funtzioa1.c

```
	gcc -I ../include KaixoMundua.c  funtzioa1.c -o KaixoMundua
```

Make

```
Hobekuntza:
KaixoMundua: KaixoMundua.o funtzioa1.o
        gcc KaixoMundua.o funtzioa1.o -o KaixoMundua
funtzioa1.o:  funtzioa1.c
        gcc -c funtzioa1.c -o funtzioa1.o
KaixoMundua.o:  KaixoMundua.c
        gcc -c -I ../include KaixoMundua.c -o KaixoMundua.o
```

Make:Aldagaiak

- Oso arrunta da aldagaiak Makefile fitxategian erabiltzea, hauekin mantenketa eta plataforma ezberdinetara edo ingurune ezberdinetara aldatzeko balio dute. Aldagaiak definitzea oso erraza da. Aldagaiaren izena (normalean letra larriz) berdin balorea, hurrengo era honetan:

  CC = gcc -O2

- Aldagaiaren edukia atzitzeko:

  $(CC) juego.c -o juego

Make:Aldagaiak

- Aldagaiak erabiltzea zabalkuntza arazoak ekar dezake, hau da, zabalkuntza errekurtziboarena:

  CC := gcc

  CC := $(CC) -O2

- “:=” erabiltzeak, “=”en ordez zabalkuntza errekurtziboaren arazoa konpontzen da.

Make: Aldagaiak

- Make barruan definitutako aldagaien adibide batzuk:
  - OBJETUAK= <objetu1><objetu2> ...
  - ITURRIAK= <iturria1><iturria2> ...

Make:Ingurune Aldagaiak

- Ere 'bash'eko ingurune aldagaiak erabili ahal ditugu. Adibidez:

```
SRC = $(HOME)/src

jokoa:

	gcc $(SCR)/*.c -o jokoa
```

Make: Aurredefinituriko Aldagaiak

'make'arentzat balio lehenetsiak dutenak:

AS: mihiztatzaile programaren izena. Lehenetsia: as

CC: C-n konpilatzeko programaren izena. Lehenetsia:gcc

CPP: g++ aurreprozesatzailearen izena. Lehenetsia $(CC) -E

CFLAG= C-ko Konpiladorearen aukerak.

RM= Fitxategiak ezabatu. Lehenetsia: rm -f (konfigurazio gabe)

Make: Aldagaiak

```
OBJETUAK=KaixoMundua.o funtzioa1.o
CFLAG= -I ../include
KaixoMundua:$(OBJETUAK)
    gcc $(OBJETUAK) -o KaixoMundua
funtzioa1.o: funtzioa1.c
    gcc -c funtzioa1.c -o funtzioa1.o
KaixoMundua.o: KaixoMundua.c funtzioa1.h
    gcc -c $(CFLAG) KaixoMundua.c -o KaixoMundua.o
```

Make: Aldagai Automatikoak

Aldagai hauek erregelaren barruan ebaluatzen dira:

$* Luzapenik gabeko objetu artxiboaren izena.

$+ Menpekotasunen artxibo guztien izenak.

$< Menpekotasunen lehen artxiboaren izena.

$? Objetua baino berriagoak diren menpekotasun artxibo guztien izenak, beraien artean tarteekin.

$@ Helburuaren izen osoa.

Make: Aldagai Automatikoak

```
CC= gcc
CFLAG= -g
OBJETUAK= geometria.o azalera.o bolumena.o
all guztia: geometria
geometria: $(OBJETUAK)
    $(CC) -o $@ $(OBJETUAK)
    ....
clean kendu:
    $(RM) geometria $(OBJETUAK)
```

Adibidea:

```
$vi define.h
#ifndef _DEFINE_H
#define _DEFINE_H
#define PI 3.1415
#endif
```

```
$vi bolumena.h
#ifndef _BOLUMENA_H
#define _BOLUMENA_H
int kubo (int x);
float esfera (int x);
#endif
```

```
$vi azalera.h
#ifndef _AZALERA_H
#define _AZALERA_H
int karratu (int x);
float zirkulu (int x);
#endif
```

Adibidea

```
$vi  azalera.c
#include "define.h"
int karratu(int x)
{return x*x;}
float zirkulu(int x)
{return PI*x*x;}
```

```
$vi bolumena.c
#include "define.h"
int kubo (int x)
{return x*x*x;}
float esfera (int x)
{return 4*PI*x*x*x/3;}
```

Adibidea

```
$vi geometria.c

#include <stdio.h>

#include "azalera.h"

#include “bolumena.h"

int main(){

int aldea,erradio;

printf(“Alde bat sartu \n");

scanf("%d",&aldea);
```

```
printf(“erradio bat sartu\n");

scanf("%d",&erradio);

printf(“Kuboa %d  da \n",kubo(aldea));

printf(" Esfera %f da \n",esfera(erradio));

printf(“Karratua %d da
\n“,karratua(aldea));

printf(“zirkulu %f  da \n",zirkulu(erradio));

return 0;

}
```

Adibidea: Liburutegi gabe

```
gcc -I. -c bolumena.c -o bolumena.o
gcc -I. -c azalera.c -o azalera.o
gcc -I. -c geometria.c -o geometria.o
gcc -I. bolumena.o azalera.o geometria.o -o
   geometria
```

Adibidea: liburutegi estatikoa

```
gcc -I. -c bolumena.c -o bolumena.o
gcc -I. -c azalera.c -o azalera.o
ar rv liborokor.a bolumena.o azalera.o
ranlib liborokor.a
gcc -I. -L. geometria.c liborokor.a -o geometria
   -static
```

Adibidea: liburutegi dinamikoa

```
gcc -I. -c bolumena.c -o bolumena.o -fPIC
gcc -I. -c azalera.c -o azalera.o -fPIC
ld -o liborokor.so bolumena.o azalera.o -shared
gcc -L. -I. -o geometria geometria.c -Bdynamic liborokor.so
//ldd geometria ->geometriak ze liburutegi dinamiko
    behar dituen aztertzen du.
//ldd dio:liborokor.so => not found
LD_LIBRARY_PATH=$LD_LIBRARY_PATH:/home/kepa/lana
export LD_LIBRARY_PATH
//ldd geometria -> geometriak ze liburutegi dinamiko
    behar dituen aztertzen du.
 ./geometria
```

Adibidea: Make

vi makefile

```
geometria:azalera.o bolumena.o geometria.o
    gcc geometria.o azalera.o bolumena.o -o
    geometria
bolumena.o: bolumena.c
    gcc -I. -c bolumena.c -o bolumena.o
azalera.o: azalera.c
    gcc -I. -c azalera.c -o azalera.o
geometria.o: geometria.c
    gcc -I. -c geometria.c -o geometria.o
```