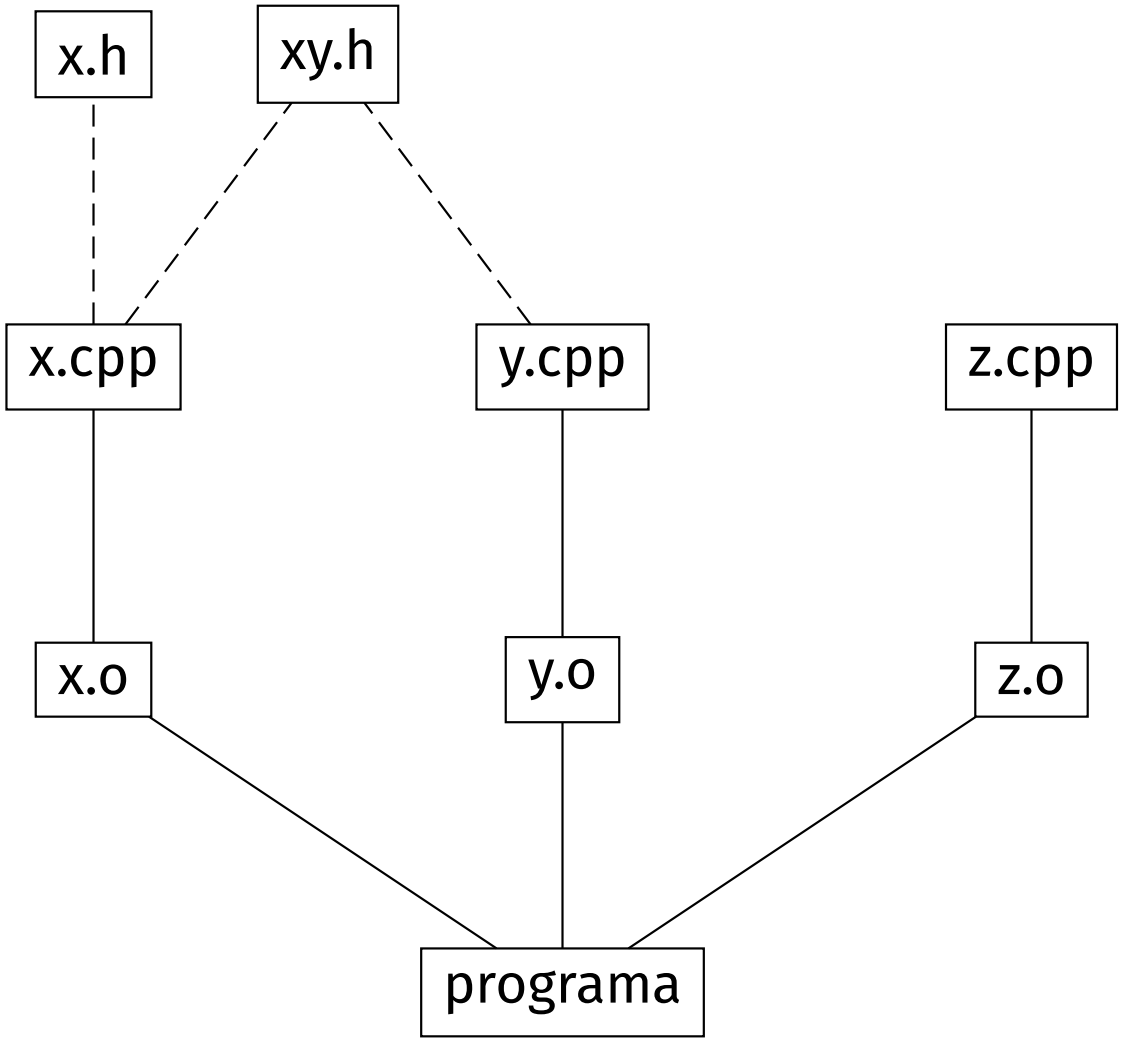
Compilación y recompilación

Francisco Palomo Lozano Inmaculada Medina Bulo

Departamento de Ingeniería Informática

Ejemplo de grafo de dependencias



Compilación del proyecto

Compilación conjunta frente a separada

1. Compilación y enlazado para obtener el ejecutable en un paso

```
c++ -o programa x.cpp y.cpp z.cpp
```

2. Compilación y enlazado por separado

```
c++ -c x.cpp y.cpp z.cpp
c++ -o programa x.o y.o z.o
```

Ventajas de la compilación separada

1. Se recompilan únicamente los módulos modificados
   - Ya sea porque lo han sido directamente
   - O indirectamente, por ejemplo, por incluir cabeceras modificadas
2. El proceso es automatizable
   - Existen varias herramientas disponibles, por ejemplo, Make
   - Make, y sus makefiles, se emplean en otras herramientas más complejas

Makefiles

Reglas

Objetivos	Ficheros a crear o actualizar si fuera necesario
Prerrequisitos	Dependencias, objetivos de los que depende otro
Recetas	Acciones a ejecutar para lograr los objetivos

Variables

1. Especiales, como `CXX`, `CPPFLAGS`, `CXXFLAGS`, `LDFLAGS`, `LDLIBS` o `RM`
2. Automáticas, como `$@`, `$<` o `$^`
3. Definidas por el usuario, como `EXE`, `OBJS`, `HDRS` o `SRCS`

Ejemplos de ejecución

1. `make` – Construye el primer objetivo disponible
2. `make CXX=clang++` – Igual, pero con el compilador indicado
3. `make all` – Construye el objetivo `all`
4. `make -n clean` – Muestra qué ejecutaría el objetivo `clean`

Makefile – Primera versión

```
# El formato de una regla es el siguiente:
#
# objetivos: prerrequisitos
#     TAB    recetas


programa: x.o y.o z.o
	c++ -o programa x.o y.o z.o

x.o: x.cpp x.h xy.h
	c++ -c x.cpp

y.o: y.cpp xy.h
	c++ -c y.cpp

z.o: z.cpp
	c++ -c z.cpp
```

Makefile – Segunda versión

```
# Con variables automáticas y especiales

CXX = c++



programa: x.o y.o z.o
	$(CXX) -o $@ $(LDFLAGS) $(LDLIBS) $^

x.o: x.cpp x.h xy.h
	$(CXX) -c $(CXXFLAGS) $<

y.o: y.cpp xy.h
	$(CXX) -c $(CXXFLAGS) $<

z.o: z.cpp
	$(CXX) -c $(CXXFLAGS) $<
```

Makefile – Tercera versión

```
# Con variables definidas por el usuario

CXX = c++
EXE = programa
OBJS = x.o y.o z.o

$(EXE): $(OBJS)
	$(CXX) -o $@ $(LDFLAGS) $(LDLIBS) $^

x.o: x.cpp x.h xy.h
	$(CXX) -c $(CXXFLAGS) $<

y.o: y.cpp xy.h
	$(CXX) -c $(CXXFLAGS) $<

z.o: z.cpp
	$(CXX) -c $(CXXFLAGS) $<
```

Makefile – Cuarta versión

```
# Con reglas implícitas

CXX = c++
EXE = programa
OBJS = x.o y.o z.o

$(EXE): $(OBJS)
        $(CXX) -o $@ $(LDFLAGS) $(LDLIBS) $^

x.o: x.h xy.h


y.o: xy.h
```

Makefile – Quinta versión

```
# Con objetivos falsos

CXX = c++
EXE = programa
OBJS = x.o y.o z.o

all: $(EXE)

$(EXE): $(OBJS)
        $(CXX) -o $@ $(LDFLAGS) $(LDLIBS) $^

x.o: x.h xy.h
y.o: xy.h

clean:
        $(RM) $(EXE) $(OBJS) core *~

.PHONY: all clean
```

Bibliografía

Gerardo Aburruzaga García
Make – Un programa para controlar la recompilación
Departamento de Ingeniería Informática. Universidad de Cádiz (2022)

Richard M. Stallman, Roland McGrath y Paul D. Smith
GNU Make – A Program for Directing Recompilation
Free Software Foundation (2023)

GNU Make Manual
https://www.gnu.org/software/make/manual
Free Software Foundation (2023)