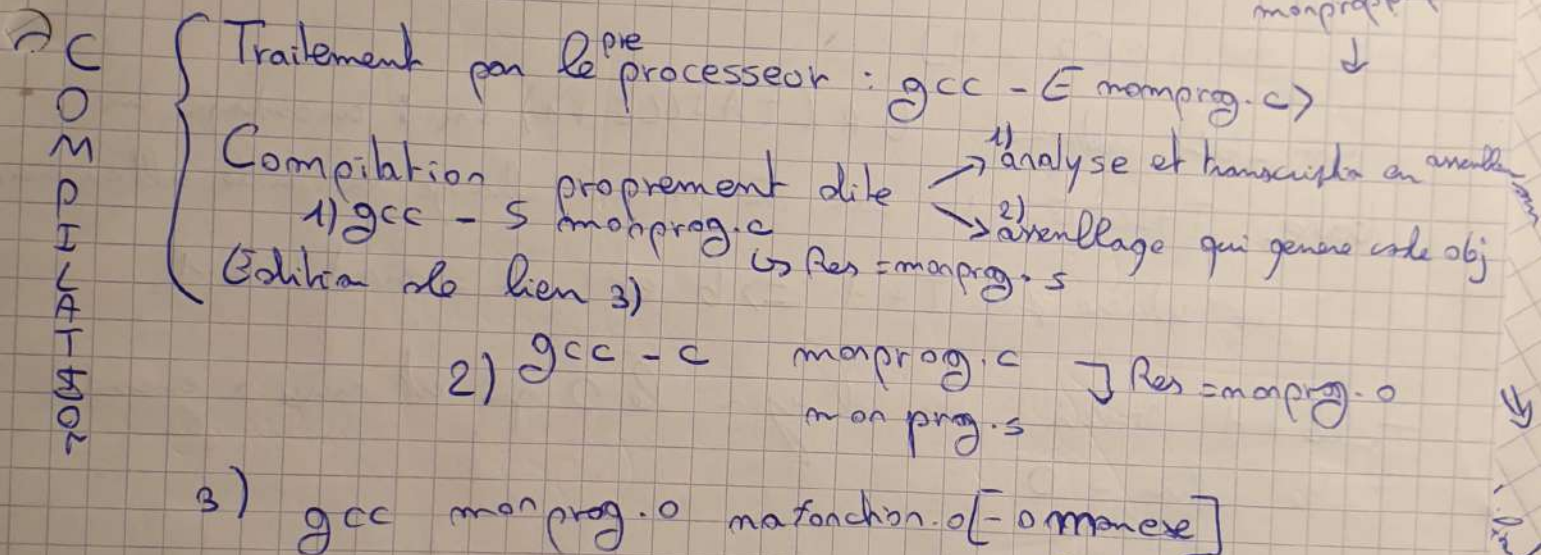


Exercice 1:

Décrire les étapes de la compilation

- 1) Analyse lexicale: conversion chaîne caractère en symbole (token) il est fait par le lexer (automate issue d'un lexème)
↓
- 2) Analyse syntaxique (consomme token) génération arbre syntaxique abstrait (table symboles continue)
↓ instance objet
- 3) Analyse sémantique
↓

Correction:



Si 3) remarque si on ne précise pas le commutateur -E, -S... alors gcc le fait lui-même par défaut

si on l'enlève l'exé devient a.out par défaut et reste quand on exéc (pas de pb si qu'un prog dans rep sinon le dernier a.out écrasera l'autre)

gcc -c monprog(2) fois

(3)

"3) exe" = undefined reference ou def multiple

TAILLE = nb de chiffre a trouver

car '0' donc Taille + 1

on minimum 3 octets

pour changer taille blocs ^{foreach} $\min(\text{taille_struct}) = ?$

#pragma pack(?)

padding

$$\text{sizeof}(mm) = \text{sizeof}(\text{max_type}) - \text{sizeof}(\text{type_x})$$

In routine retourner $b = nb$ chiffre bien places
— mal —

avec $m, b \in [0..8]$

mon_text retourne $b * (\text{TAILLE} + 1) + m$: encodage en

base TAILLE + 1 car on ne change la taille ultérieurement

div entière par $T + 1 \Rightarrow b$ et $m = \text{mod } T + 1$

Taille + 1 valeur car il y a le 0 qui est aussi a encodé
Taille char + 0

CC : encodera le compilateur que l'on veut utiliser

ainsi en changeant CC on changera H ses occurrences

clean : ^{cible} tab
→ m & 0 mon-exe | nettoie tab les 0 car clean
renvoie m rien

make clean

en haut mk file

all: chat serveur ^{tab} dernier
cible si question man make

dans un m dossier on peut avoir 2 Makefile un avec maj Makefile
devant (et l'autre : makefile) on l'un et prioritaire sur
l'autre.