

A thick dark blue vertical bar runs down the left side of the page. A blue arrow points to the right from this bar, containing the date.

29/09/2020

# CER Prosit 3

Compilateur

Several thin, curved lines in dark blue and light grey originate from the bottom left and sweep upwards and to the right.

Robin HATIER  
CESI NICE

## Synthèse du prosit

---

Ce prosit m'a permis de découvrir les différentes étapes de la compilation. J'ai aussi approfondi mes connaissances sur l'os linux.

## Table des matières

---

Synthèse du prosit.....	1
Contexte .....	1
Objectif.....	2
Contraintes.....	2
Plan d'action.....	2
Réalisation du plan d'action .....	3
<b>Partie 1 : Mots clefs .....</b>	<b>3</b>
<b>Partie 2 : Voir les notions importantes (automates, compilation, code C...) .....</b>	<b>4</b>
<b>Partie 3 : Coder un code simple en C .....</b>	<b>9</b>
<b>Partie 4 : Installer le compilateur sur l'OS Linux .....</b>	<b>10</b>
<b>Partie 5 : Compiler le code C en langage binaire .....</b>	<b>10</b>
Conclusion et retour sur les objectifs : .....	11
Références des méthodes et outils utilisés : .....	11
Les références : .....	11

## Contexte

---

L'entreprise THS nous missionne pour récupérer et traiter les informations météorologiques des capteurs.

## Objectif

---

Comment utiliser un compilateur pour passer d'un langage haut niveau au binaire ?

## Contraintes

---

- Compilateur gcc
- Utilisation de langage bas niveau
- Utiliser plusieurs fichiers de code
- Automatiser la phase de téléversement, le numéro de série, la mise en debug, uat et release
- Documentation ATMEL
- Compilateur gcc sous linux

## Plan d'action

---

### Théorique

Partie 1 : Mots clefs

Partie 2 : Voir les notions importantes (automates, compilation, code C)

### Pratique

Partie 3 : Coder un code simple en C

Partie 4 : Installer le compilateur sur l'OS Linux

Partie 5 : Compiler le code C en langage binaire

# Réalisation du plan d'action

---

## Partie 1 : Mots clefs

- GCC est un produit diffusé sous licence libre et amélioré/maintenu par une importante communauté. Il est disponible sur toutes les plateformes du marché : Mac, Linux, Windows et d'autres. C'est un 'cross-compiler', c'est à dire qu'il est capable de générer du code pour la machine sur laquelle il est exécuté, mais aussi pour beaucoup d'autres plateformes, y compris les processeurs Atmel qui équipent nos Arduino.
- UAT : C'est un test de validation informatique qui permet de vérifier si toutes les exigences client, décrites dans le document de spécification du logiciel, sont respectées.
- ATMEL : Atmel est un fabricant mondial de composants à semi-conducteur.
- AVR gcc : AVR-GCC est un compilateur qui prend du code de haut niveau en langage C et crée une source binaire qui peut être téléchargée dans un microcontrôleur AVR.
- Debug : Dans la programmation informatique et le développement de logiciels, le débogage est le processus de recherche et de résolution de bogues dans des programmes informatiques, des logiciels ou des systèmes.
- Phase de téléversement : Le téléversement renvoie au fait de transmettre des données de votre ordinateur vers récepteur (un arduino par exemple).
- Automate : Un automate à états finis (AF) est un modèle d'un système et de son évolution, c'est-à-dire une description formelle du système et de la manière dont il se comporte.
- Automate fini déterministe : Un automate fini déterministe, parfois abrégé en AFD est un automate fini dont les transitions à partir de chaque état sont déterminées de façon unique par le symbole d'entrée.
- Code bas/haut niveau : Le langage bas niveau est un programme lisible uniquement par la machine. Alors que le langage haut niveau sera lisible par

l'homme. Les langages bas niveau sont difficiles à écrire et à compiler, mais les langages haut niveau sont faciles à écrire et à compiler.

- Binaire : Le système binaire est un système d'écriture informatique en base 2. Il n'utilise que deux caractères : le 0 et le 1. Dans le vocabulaire de la numérotation binaire, on appelle « bit » les chiffres, qui ne peuvent prendre que ces deux valeurs.

Partie 2 : Voir les notions importantes (automates, compilation, code C...)

### Partie code C :

Un programme en C se compose de trois parties :

- la **partie** déclaration des variables (variables globales)
- la **partie** initialisation et configuration des entrées/sorties : la fonction `setup()`
- la **partie** principale qui s'exécute en boucle : la fonction `loop()`

```
1 int brocheCapteur = A0;    // selection de la broche sur laquelle est connectée le capteur
  int brocheLED = 13;       // selection de la broche sur laquelle est connectée la LED
  int valeurCapteur = 0;    // variable stockant la valeur du signal reçu du capteur

2 void setup() {
  // broche de la LED configurée en sortie
  pinMode(ledPin, OUTPUT);
}

3 void loop() {
  // lecture du signal du capteur
  valeurCapteur = analogRead(brocheCapteur);
  // allume la LED
  digitalWrite(brocheLED, HIGH);
  // delai de "valeurCapteur" millisecondes
  delay(valeurCapteur);
  // éteint la LED
  digitalWrite(brocheLED, LOW);
  // delai de "valeurCapteur" millisecondes
  delay(valeurCapteur);
}
```

Tableau à connaître pour coder en C :

Langage C	Traduction	Utilisation
<b>Int</b>	Nombre entier	Insérer une variable
<b>Float</b>	Nombre décimal	Insérer une variable
<b>Const int</b>	Constante	Insérer une constante
<b>If</b>	Si	Début condition
<b>Else</b>	Sinon	Fin condition
<b>Else if</b>	Sinon si	Dans une condition
<b>Switch</b>	Selon	Début condition
<b>Break</b>	Alors	Fin condition
<b>For</b>	Pendant	Boucle
<b>While</b>	Tant que	Boucle
<b>Do...while</b>	Jusqu'à ce que	Boucle
<b>Goto</b>	Va à	Permet de se rendre à l'étiquette
<b>Etiquette :</b>	etiquette	Permet au goto de savoir où aller
<b>Input</b>	Entrée	Déterminer une entrée
<b>Output</b>	Sortie	Déterminer une sortie
<b>Println</b>	Afficher en sautant ligne	Écrire qlqc
<b>pinMode</b>	Famille de composant	Output ou Input
<b>DigitalWrite</b>	« Ecrire » sur un PIN	Envoyer une info
<b>DigitalRead</b>	« Lire » sur un PIN	Recevoir une info
<b>Delay()</b>	Délais	Attendre
<b>=</b>	Égale	Affectation
<b>==</b>	Ça avec ça	Comparaison
<b>&amp;&amp;</b>	ET logique	ET
<b>  </b>	OU logique	OU ( options+sifht+L)
<b>;</b>	Fin d'une instruction	Chaque ligne
<b>{ }</b>	Début fin instruction	Avant boucle, condition
<b>HIGH</b>	Activé (inversé PullUp)	Lampe, Bouton
<b>LOW</b>	Pas Activé (inversé PullUp)	Lampe, Bouton

## Partie assembleur :

Voir prosit précédent

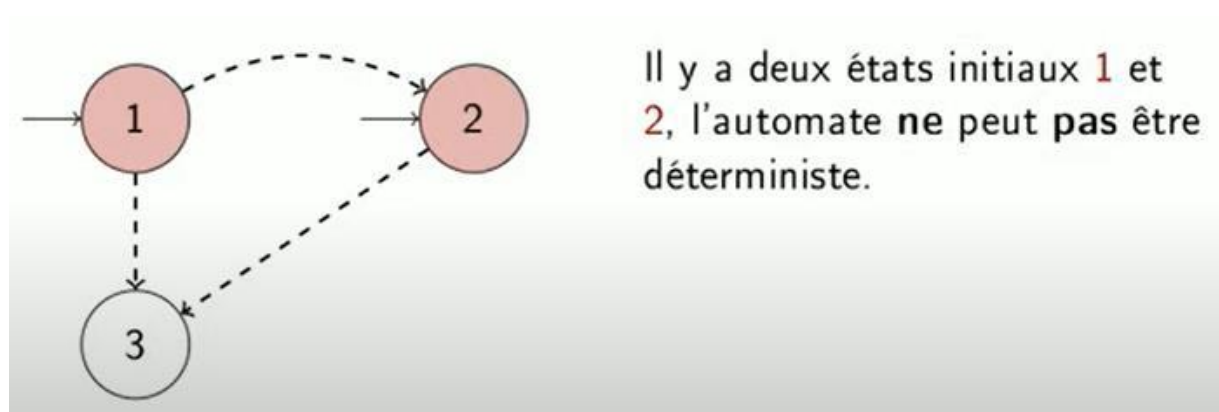
## Partie automates :

Un automate à états finis (AF) est un modèle d'un système et de son évolution, c'est-à-dire une description formelle du système et de la manière dont il se comporte.

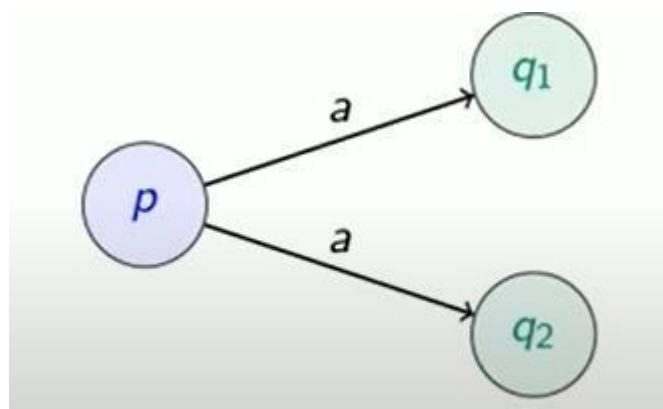
Un AF est composé d'un ensemble fini d'états (représentés graphiquement par des cercles), d'une fonction de transition décrivant l'action qui permet de passer d'un état à l'autre (ce sont les flèches), d'un ou plusieurs états initiaux (désignés par des flèche entrantes), d'un ou plusieurs états finaux (désignés par le symbole). Un AF est donc un graph orienté où les nœuds correspondent aux états et les arcs aux flèches.

Un automate fini est dit déterministe s'il vérifie les **deux conditions** suivantes :

- Il possède exactement **un** état initial (état initial représenté par une petite flèche à côté des ensembles finis):



- Au maximum **UNE** transition qui part de p édicté par a (ou b ou c...) . Max une flèche par lettre :



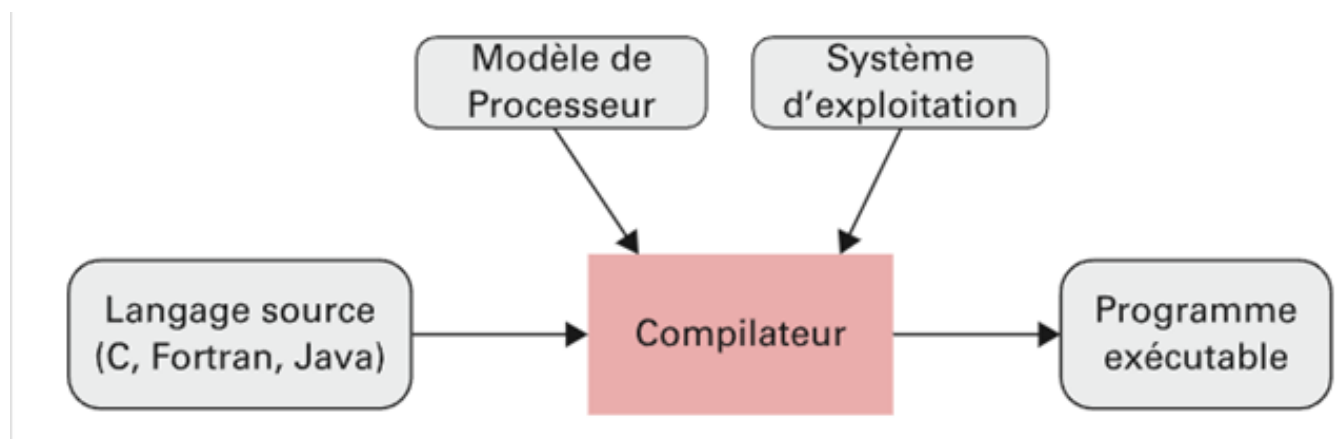
## Partie compilateur :

Compilateur : L'usage habituel d'un compilateur est de rendre exécutable un programme rédigé dans un langage de programmation source (comme le C) vers un programme exécutable par un processeur et un système d'exploitation spécifiques.

D'un point de vue plus large, la compilation est une des étapes dans le cycle de vie d'un programme. Cette figure indique les différents états (Idée, Algorithme, Programme source puis binaire, Exécutable, en mémoire puis en cours de fonctionnement) vers l'exécution d'un programme et, au-dessus les différents acteurs impliqués dans cette création (le designer, le programmeur et l'utilisateur).

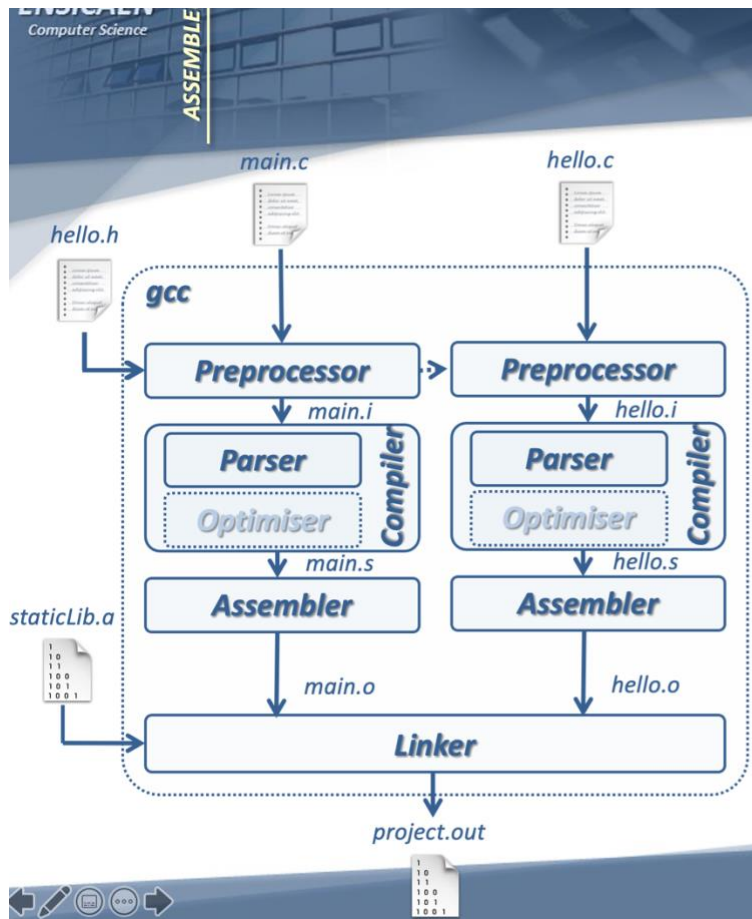
Le compilateur a pour objectifs de rendre exécutable le programme source :

- en le transformant vers le langage binaire d'un processeur donné (Intel, ARM, etc.) ;
- en l'optimisant pour une version particulière d'un processeur ;
- en liant les binaires avec le système d'exploitation de la machine (Windows, Linux, Android, etc.).





Voici les différentes étapes de la compilation :



- **Préprocesseur :** interprétation directives compilation (#) et suppression commentaires
- **Compilateur :** Analyse programme (lexicale, syntaxique, sémantique..), code intermédiaire, optimisation optionnelle, génération ASM (CPU architecture dépendant).
- **Assembleur :** Génération code binaire/objet relogeable pour CPU cible
- **Editeur de liens :** Liens entre fichiers objets et bibliothèques (statique et/ou dynamique). Génération et chargement code binaire/exécutable absolu

## Pratique

### Partie 3 : Coder un code simple en C

```
const int lum = A0;
int mesure_lum ;
const int buttonPin = 3;
int buttonState = 0;
//int mesure_temp;
//int degres_mesure;
//const int temp = A3;

void setup()
{
  Serial.begin(9600);
  pinMode(lum, INPUT);
  pinMode(buttonPin, INPUT);
  //pinMode(temp, INPUT);
}

void loop()
{
  Serial.print("Luminosite :");
  mesure_lum = analogRead(lum);
  Serial.println(mesure_lum);

  buttonState=digitalRead(buttonPin);

  if (buttonState == HIGH) {

    Serial.println("Bouton non allume");
    delay(1000);
  }
  else {

    Serial.println("Bouton allume");
    delay(1000); }

  /*Serial.print("Temperature :");
  mesure_temp = analogRead(temp);
  degres_mesure = ((mesure_temp ) / 10) ;
  Serial.println(degres_mesure);*/
}
```

J'ai codé un programme simple pour le compiler par la suite.

Ce code fait intervenir un capteur de luminosité ambiante, des boutons et le moniteur de série pour afficher des sorties en fonctions des boutons.

Toutes les secondes le capteur de luminosité envoie une valeur sur le moniteur de série, et nous affichons l'état du bouton.

- Avant le setup j'ai déclaré les variables constantes qui correspondent au PIN utilisé. J'ai aussi déclaré les variables qui permettront de stocker l'état du bouton et la valeur captée par le capteur de luminosité.
- Ensuite dans le setup j'ai initialisé le moniteur de série et j'ai déclaré les INPUT qui correspondent aux boutons et au capteur
- Pour finir dans la loop nous avons le code permettant de récupérer la valeur du capteur et l'état du bouton, pour les print d'une belle façon sur le moniteur.

## Partie 4 : Installer le compilateur sur l'OS Linux

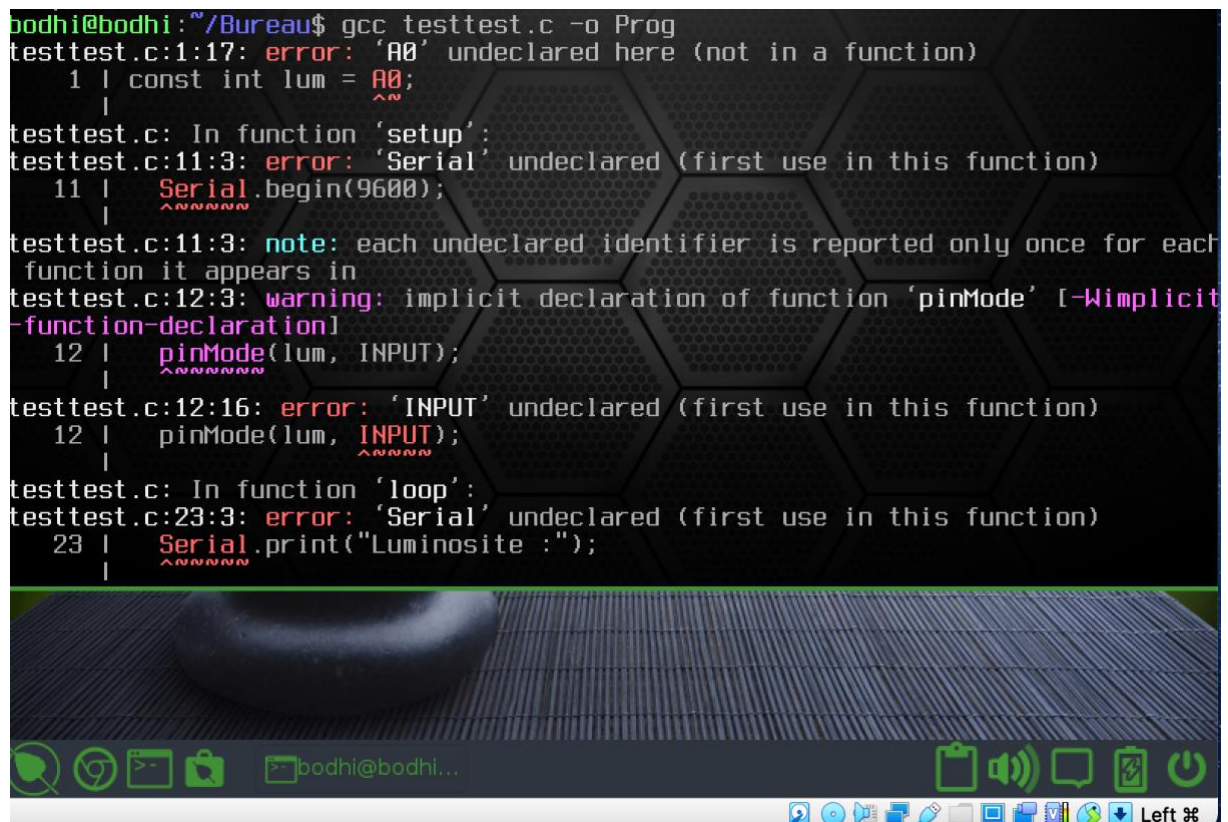
Nous avons décidé d'utiliser le compilateur GCC sur Linux.

Pour l'utiliser nous devons installer l'apt grâce à une ligne de commande. [ICI](#)

```
apt-get install build-essential
```

## Partie 5 : Compiler le code C en langage binaire

En compilant mon programme en C grâce au GCC des messages d'erreurs apparaissent et je n'obtiens pas mon fichier compiler en langage de bas niveau.



```
bodhi@bodhi:~/Bureau$ gcc testtest.c -o Prog
testtest.c:1:17: error: 'A0' undeclared here (not in a function)
  1 | const int lum = A0;
    |                 ^~
testtest.c: In function 'setup':
testtest.c:11:3: error: 'Serial' undeclared (first use in this function)
  11 |     Serial.begin(9600);
    |     ^~~~~~
testtest.c:11:3: note: each undeclared identifier is reported only once for each
function it appears in
testtest.c:12:3: warning: implicit declaration of function 'pinMode' [-Wimplicit-
function-declaration]
  12 |     pinMode(lum, INPUT);
    |     ^~~~~~
testtest.c:12:16: error: 'INPUT' undeclared (first use in this function)
  12 |     pinMode(lum, INPUT);
    |                 ^~~~~~
testtest.c: In function 'loop':
testtest.c:23:3: error: 'Serial' undeclared (first use in this function)
  23 |     Serial.print("Luminosite :");
    |     ^~~~~~
```

## Conclusion et retour sur les objectifs :

---

Nous pouvons utiliser GCC pour compiler nos programmes, cependant ce n'est pas le plus pratique quand nous utilisons l'IDE arduino. En effet, l'IDE Arduino aurait pu nous compiler directement le programme et le téléverser. Cependant ce projet m'a permis de découvrir le fonctionnement et les étapes de la compilation .

## Références des méthodes et outils utilisés :

---

J'ai utilisé le réseau internet afin d'exploiter les ressources mises à notre disposition et pour me renseigner sur la compilation. J'ai aussi utilisé Word, pour rédiger ce rapport. Et une carte arduino et différents capteurs

## Les références :

---

<https://pub.phyks.me/sdz/sdz/compilez-sous-gnu-linux.html>

[https://www.canal-u.tv/video/centre\\_d\\_enseignement\\_multimedia\\_universitaire\\_c\\_e\\_m\\_u/architecture\\_et\\_technologie\\_des\\_ordinateurs\\_langage\\_d\\_assemblage.12509](https://www.canal-u.tv/video/centre_d_enseignement_multimedia_universitaire_c_e_m_u/architecture_et_technologie_des_ordinateurs_langage_d_assemblage.12509)

<https://c.developpez.com/faq/?page=Techniques-de-compilation-et-d-edition-de-liens#Qu-est-ce-que-le-compileur>