Planning pour le module R1.03 (Architecture)

D. Roegel

26 août 2024

1 Volume

14h TD + 8h TP + 2h DS
Chaque semaine, en principe : 4h de TD + 2h de TP en 1/2 groupes

2 Documents fournis

À part le présent document, il n'y a que le fichier du cours qui est en

`https://tmpdocs.github.io/r103/r103-cours.pdf`

Ce document est susceptible d'évoluer encore un petit peu, mais il est en principe presque finalisé.

Le lien est à donner à vos étudiants. Je ne mets rien sur ARCHE. On peut éventuellement y laisser les supports de l'ancien cours, à titre d'information ou de complément.

Il n'y a pas de fichiers séparés pour les TDs, TPs, ni même de transparents de cours. Tout est en un. Projetez le support et agrandissez si nécessaire.

3 Évaluation

Il y a un DS à la fin, plus une note de participation (0 à 5), plus une évaluation orale au milieu du module (5mn sur un exercice).

Le DS sera composé d'exercices, de questions de cours et de QCM ; il faudra donc avoir lu tout le cours. Aucun document ne sera autorisé au DS, sauf éventuellement deux pages manuscrites, et les sacs devront être rassemblés à l'avant des salles d'examen (avec les téléphones).

Le plus simple sera de faire l'examen oral pendant le TP suivant le chapitre 5. Vous serez dans votre bureau, vous faites venir les étudiants toutes les 5mn, vous leur donnez un sujet (que je vous donnerai, genre IEEE754) et ils vous l'exposent en 5mn, sans préparation. Vous donnerez une note entre 0 et 5. Cette note sera un élément de la note finale (on verra comment on combine les notes).

4 Planning détaillé

Je souhaite que l'on fasse deux ou trois chapitres par semaine :
— Semaine 1 : chapitres 1, 2, 3 et 10
— Semaine 2 : chapitres 4 et 5
— Semaine 3 : chapitres 6 et 7
— Semaine 4 : chapitres 8 et 9

J'ai voulu fournir beaucoup de cours pour que les étudiants aient un document de référence (notamment pour le C). Maintenant, il ne s'agit pas de tout faire ! À vous de sélectionner. Notamment pour les exercices. L'idée est de faire lire le cours d'avance par les étudiants et ensuite de répondre aux questions. Il s'agit aussi de les faire faire des exercices variés. Les plus rapides pourront en faire plus. Je ne fournis pas de solutions. Si les étudiants souhaitent une solution, qu'ils demandent à ChatGPT.

Il est important de rester en phase avec le planning. Si un chapitre n'est pas fini, c'est aux étudiants de finir de le lire, vous passez tout de même au chapitre suivant. Par contre, si à la fin du cours les étudiants veulent des explications complémentaires, prenez le temps de les leur donner.

À part au premier TD, il faut se limiter à commenter les figures et photographies et à répondre aux questions. Ne faites pas plus d'une heure de cours en deux heures, et fractionnez votre présentation.

Pour les exercices, faites en priorité ceux marqués d'un astérisque, ensuite vous faites comme vous voulez/pouvez. Imposez cependant un ordre, sinon cela partira dans tous les sens. N'hésitez pas, si vous voulez, à utiliser vous-même ChatGPT pour les corrections. Je n'en fournis pas. Par ailleurs, le plus important n'est pas toujours la correction, mais le fait d'avoir des idées pour résoudre un problème.

En pratique :

TD1 : vous donnez le lien vers le support, vous faites une introduction générale au cours (les différents chapitres), vous parcourez avec les étudiants les chapitres 1 et 2, avec les compléments historiques ; et vous passez à des exercices le plus vite possible ; faites le moins de cours possible, à part à la première séance. Essayez de vous limiter aux exemples, à la description des figures et des images. Lors du TD1, introduisez le C avec le chapitre 10 (seulement le minimum pour compiler et les bases) ; évidemment, linux est mis en avant, si les étudiants veulent utiliser autre chose, on ne pourra pas forcément les aider ; si vous n'avez pas fini, demandez aux étudiants de lire jusqu'à la fin, vous pourrez toujours y revenir plus tard.

Enfin, demandez aux étudiants de lire le chapitre 3.

TD2 : repassez rapidement en revue le chapitre 3 que les étudiants doivent avoir lu et répondez à leurs questions ; puis passez à des exercices (je vous laisse choisir) ; demandez aux étudiants de lire le chapitre 4.

TP : faites des exercices faisant appel au C ou à d'autres langages, au choix. demandez aux étudiants de faire chez eux les autres exercices.

TD3 : passez en revue le chapitre 4, faites des exercices et demandez de lire le chapitre 5.

TP : idem au précédent

TD4 : passez en revue le chapitre 5, faites des exercices et demandez de lire le chapitre 6.

TP : idem au précédent

TD5 : passez en revue le chapitre 6, faites des exercices et demandez de lire le chapitre 7.

TP : exercices à faire par les étudiants + INTERROGATION ORALE de 5mn

TD6 : passez en revue le chapitre 7, faites des exercices et demandez de lire le chapitre 8.

TP : idem au précédent

TD7 : passez en revue le chapitre 8, faites des exercices et demandez de lire le chapitre 9.

TP : idem au précédent

5 Devoirs à rendre

Je souhaite que vous demandiez à vos étudiants de rendre certains exercices (au moins un par séance d'exercices). Vous pouvez leur demander de le rendre sous forme papier, ou par un autre moyen. En ce qui me concerne, je demande de rendre les travaux via *github classroom*. Cela initiera les étudiants à *github classroom* qu'ils auront l'occasion de revoir par la suite.

Je donne quelques indications sur *github classroom* ci-dessous.

À partir des devoirs rendus, je souhaite avoir une note globale entre 0 et 5 à la fin du module. Quelqu'un qui n'a rien rendu a 0.

De mon côté, je vais demander aux étudiants de me rendre tous les exercices, ce qui les forcera à les rédiger.

6 Github classroom

Github classroom est un moyen simple de gérer des classes et des devoirs. Voici quelques indications si vous voulez l'utiliser :

1. allez sur `https://classroom.github.com`
2. choisissez « Sign in » et loggez-vous avec un compte github (ou créez un compte github si vous n'en avez pas). On arrive à la page « Your Classrooms ».
3. Créez une classe pour votre cours en cliquant sur « New classroom ».
   (a) Pour ce faire, vous devez aussi créer une organisation, car les classroom sont regroupés par organisation. Les noms des organisations sont uniques, vous ne pouvez donc pas tous prendre le même. Vous pouvez prendre votre nom suivi d'un numéro, ou quelque chose comme cela, ou y inclure « architecture ». C'est comme vous voulez.
   (b) Une fois l'organisation créée et/ou choisie, vous pouvez choisir le nom de la classroom ; moi, j'ai pris architecture2024

4. Ensuite, github vous demande de lier d'autres enseignants, mais sautez cette étape.
5. Ensuite, il faut donner la liste de vos étudiants. C'est mieux de le faire ici, sinon vous aurez des identifiants obscurs d'étudiants dans les dépôts. Le plus simple est de récupérer les mails de vos étudiants, de les mettre dans un fichier texte et de les donner à github pour qu'il constitue la classe.

   En attendant, vous pouvez continuer sans étudiants et les ajouter après le premier TD, quand vous aurez les mails. Il suffit pour cela d'aller sur la page de la classe, de cliquer sur « Students » et, dans la case à droite, de donner votre liste, à raison d'un mail par ligne. Vous pouvez aussi, si vous ne voulez pas récupérer les mails, mettre simplement les noms des étudiants. Moi, je préfère mettre les mails.
6. Une fois ceci fait, vous pouvez créer un *assignment*, et il y aura un dépôt par étudiant. Il s'agit de créer un seul *assignment*, pas un *assignement* par exercice. Chaque étudiant aura un dépôt où il pourra mettre certains exercices que vous demanderez de rendre (chaque enseignant gère cela indépendamment).

   Pour ma part, j'ai appelé cet *assignment* « Exercices architecture ». Une fois créé, vous récupérez un lien que vous donnez aux étudiants. Ceux-ci se connectent et trouvent leur nom ou mail et se raccrochent à votre classe.

Tout cela a peut-être l'air un peu compliqué, mais en fait c'est assez simple et bien fait. Si vous avez commis une erreur, vous pouvez supprimer un dépôt, un *assignment* ou une classe.

Après cela, il vous suffira de temps en temps de regarder les dépôts, vous n'aurez plus rien à créer.

Du côté des étudiants, ils doivent attendre le lien que vous leur envoyez (ou notez au tableau). À eux de créer un compte github s'ils n'en ont pas. Ensuite, le plus simple est de mettre l'exercice dans un fichier `ex-X-Y.txt` (ou `.c`) où X est le chapitre et Y le numéro de l'exercice, et d'uploader ce fichier. Il y a des vidéos pour cela sur *youtube*.

7 Indications pour les illustrations

Voici encore quelques indications pour les illustrations du cours :

> attention, vérifier qu'il n'y a pas de décalage
> de numéro, j'ai au moins ajouté une figure quelque part

— les figures manuscrites sont toutes décrites dans le texte ; si nécessaire, je peux encore ajouter des précisions, mais je pense que c'est assez clair ;
— pour les photographies « historiques », voici ce que vous pouvez éventuellement dire en cours (vous n'êtes pas obligé de toutes les décrire, vous n'aurez sans doute pas le temps) :
  — fig 1.3 : Intel 4004, il me semble bon que les étudiants aient au moins une idée de la date (1971)

— fig 1.4 : von Neumann, dites que c'était un mathématicien qui a touché à beaucoup de domaines
— fig 1.5 : Harvard Mark I, notez la taille
— fig 1.7 : attirez l'attention sur quelques composants comme l'UAL, le registre d'instructions, etc. ; et dites que le 6502 est le processeur qui a lancé Apple
— fig 1.8 : Babbage, automatisation de calculs de tables par la méthode des différences finies, puis machine analytique
— fig 1.9 : fragment de la première machine à différences
— fig 1.10 : fragment de la machine analytique (jamais achevée, mais il y a actuellement un projet pour le faire)
— fig 1.11 : machine analytique de Ludgage (jamais construite), les petites tiges servaient à stocker les valeurs
— fig 1.12 : Torres Quevedo, a aussi conçu une machine analytique, un automate pour les échecs, et plein d'autres choses
— fig 1.13 : analyseur différentiel de Bush, on est ici dans le calcul analogique
— fig 1.14 : machine Z3 de Zuse, ordinateur à relais
— fig 1.15 : machine à prédire les marées, exemple de machine qui construit une fonction complexe comme somme de fonctions sinusoïdales
— fig 1.16 : un calculateur analogique de 1949, insister sur la différence entre analogique et numérique (ou digital, comme les gens disent aujourd'hui)
— fig 1.17 : vous pouvez expliquer le fonctionnement d'un relais (pas très difficile !)
— fig 1.18 : ce à quoi ressemblaient les relais de Zuse
— fig 1.19 : exemple de lampe triode, le filament chauffé envoie des électrons qui sont recuillis par la plaque, mais le courant qui va du filament à la plaque est amplifié ou réduit par le potentiel de la grille ; la lampe triode sert donc d'amplificateur ;
— fig 1.20 : une lampe soviétique !
— fig 1.21 : des lampes dans l'ENIAC ; elles tombaient régulièrement en panne ;
— fig 1.22 : un module de l'IBM 705 avec ses lampes ;
— fig 1.23 : des transistors
— fig 1.24 : encore des transistors (celui du milieu en haut est le fameux CK703)
— fig 1.25 : le die du 8086 avec 29000 transistors
— fig 1.26 : idem, mais on peut zoomer pour voir les transistors
— fig 2.3 : montrer la lecture de ces tables
— fig 2.6 : carte perforée 80 colonnes, montrer comment ça se lit
— fig 2.9 : métiers de Jacquard, montrer les cartes perforées qui programment les motifs de tissage
— fig 2.10 : je vous laisser décoder
— fig 3.1 : les puissances de 2 chez Leibniz vers 1703
— fig 3.6 : les triplets pythagoriciens de Plimton 322
— fig 4.2 : l'ordinateur d'Apollo a beaucoup fait avancer les choses et on a ici un petit élément de cet ordinateur
— fig 4.22 : Aristote, parler un peu des origines de la logique (des syllogismes, genre Tous les hommes sont mortels, Socrate est un homme, donc Socrate est mortel) (c'est un syllogisme Barbara, cf. l'un des exercices)

— fig 5.11 : règle à calcul, vous pouvez expliquer le calcul de sqrt(2.1) ;
— fig 5.16 : les volumes de Knuth
— fig 5.17 : la vraie machine d'Anticythère (i.e., pas celle d'Indiana Jones)
— fig 5.18 : Pascaline, pour faire des additions avec un stylet (comme les anciens téléphones S63)
— fig 5.19 : machine de Leibniz, aussi pour des multiplications (on pouvait décaler la valeur à multiplier)
— fig 5.20 : arithmomètre Thomas, très courant à la fin du 19e siècle
— fig 5.21 : machine Odhner, ultra courante dans tous les bureaux jusque vers 1960 ; (un tour de manivelle = une addition)
— fig 5.22 : une machine Friden, je crois le modèle qui pouvait calculer les racines carrées (à vérifier)
— fig 5.23 : la célèbre calculatrice de poche Curta
— fig 6.9 : L'IBM PC, un peu oublié aujourd'hui, mais révolutionnaire à l'époque
— fig 6.10 : L'un des ancêtres des processeurs RISC
— fig 6.11 : un programme tout numérique
— fig 6.12 : un programme pour IBM 650, vraiment de l'assembleur
— fig 6.13 : exemple en FORTRAN, vous pouvez expliquer le code
— fig 7.6 : hiérarchie mémoire de l'i7, montrer simplement qu'il y a plusieurs caches et la mémoire principale
— fig 7.7 : tambour de mémoire, qui tourne grâce au moteur de droite !
— fig 7.8 : un module de 1024 core, je pense construit à la main ! voyez ici : `https://www.youtube.com/watch?v=paQ3zIsz1-8` c'est tout petit, chaque petit tore stocke un bit ;
— fig 7.25 : les trois anciens formats de disquettes, en haut 8.5" (que je n'ai vu que de loin), en bas à droite 5"1/4 (que j'ai beaucoup utilisé), puis 3"1/2, que nous avons tous connus ;
— fig 7.31 : les armoires avec une bande qui se déroule d'un disque à l'autre, comme les cassettes audio (les étudiants connaissent-ils les cassettes audio ?)
— fig 7.33 : programmation de l'ENIAC en déplaçant manuellement des fiches
— fig 9.3 : l'UNIVAC 1 dans sa salle toute propre et tous ses accessoires
— fig 9.4 : là où on entrait un programme dans l'IBM 650, pas d'écran...
— fig 9.5 : imprimante matricielle
— fig 9.6 : interface de l'ordinateur (embarqué) d'Apollo ; c'était minimaliste (AGC = Apollo Guidance Computer)
— fig 10.2 : les auteurs du C, devant le PDP-11, sans écran