# Partiel de INF104, 18/06/2015

1h30 – Sans documents

## *Questions sur le langage C (5 points)*

**Question 1 (1 point) :** On suppose qu'on a créé un fichier exécutable appelé `display_cl` à partir du code C fournit ci-dessous.

```
#include <stdio.h>

int main(int argc, char* argv[]){
  printf("value of argc: %d\n", argc);
  int i;
  for(i=0;i<argc;i++)
  {
    printf("value of argv[%d]: %s\n", i, argv[i]);
  }
  return 0;
}
```

On exécute la commande : `./display_cl –c –d 5`

Quel affichage obtient-on ? Justifier votre réponse en précisant les valeurs prises par les paramètres argc et argv.

**Question 2 (4 points) :** Ecrire le code C qui implémente la fonction **strcpy**. Pour information, la fonction **`strcpy`** copie la chaine de caractères `src` dans la chaine de caractère `dst` (y compris le caractère de fin de chaine : '\0').

Il faut donc que vous complétiez le code ci dessous (en remplaçant les ...) :

```
void strcpy(char *dst, const char *src)
{
  ...
}
```

## *Questions sur la chaîne de production (4 points)*

**Question 3 (2 points):** Expliquez brièvement la différence entre la compilation et l'édition de lien. Quelle option de **`gcc`** permet de s'arrêter avant l'édition de lien ?

**Question 4 (2 points):** on souhaite écrire un makefile qui compile un programme C à partir de trois fichiers C : main.c, graph.c, path_computation.c. Compléter le makefile suivant en remplaçant les pointillés par le code qui convient.

```
########################################
CC       = gcc
CFLAGS   = …
LD       = gcc
OFILES   = main.o graph.o path_computation.o
########################################

clean:
        rm *.o prog

all: prog

#production de l'objet main.o (a completer)
main.o: …
        $(CC) $(CFLAGS) …

# production de l'objet graph.o (a completer)
graph.o: …
        $(CC) $(CFLAGS) …

# production de l'objet path_computation.o (a completer)
path_computation.o: …
        $(CC) $(CFLAGS) …


# production de l'executable prog (a completer)
prog: $(OFILES)
        $(LD) … -o prog
```

## *Questions sur l'ordonnancement (4 points)*

On considère l'exécution de trois processus P1, P2 et P3 ayant les caractéristiques suivantes :

- P1 démarre à la date T=0 seconde, P2 démarre à la date T=1 seconde, et P3 démarre à l'instante T=2 secondes.
- P1 dure 3 secondes, P2 dure 5 secondes et P3 dure 10 secondes.

**Question 5 (2 points) :** Nous avons représenté sous la forme d'un chronogramme l'exécution de ces processus. Sur ce chronogramme, chaque case représente un intervalle de temps de une seconde. Une case marquée d'un X signifie que le processus est dans l'état « prêt » alors qu'une case noire signifie que le processus est dans l'état « actif ». Donner le nom **des deux** politiques d'ordonnancement (vues en cours) permettant d'obtenir cette exécution quelles que soient les priorités de P1, P2 et P3.

| | | | | | | | | | | | | | | | | | |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| P1 | ■ | ■ | ■ | | | | | | | | | | | | | | |
| P2 | | X | X | ■ | ■ | ■ | ■ | ■ | | | | | | | | | |
| P3 | | | X | X | X | X | X | X | ■ | ■ | ■ | ■ | ■ | ■ | ■ | ■ | ■ |

**Question 6 (2 points) :** En suivant la même convention de notation que pour le chronogramme de la question précédente, compléter le chronogramme **fourni en annexe**, pour représenter l'exécution de ces processus : on suppose ici que la politique d'ordonnancement est « **tourniquet par niveau de priorité** » (ou *« Round Robin within priority »*) ; que le quantum de temps associé à cette politique est de 1 seconde ; et que la priorité de P1 est égale à la priorité de P2 et supérieure à la priorité de P3.

## *Questions sur la pagination de la mémoire (3 points)*

**Question 7 (1 point) :** en pagination, que signifie LRU ? Donnez sa définition et expliquez dans quel type de situation LRU est utilisé.

**Question 8 (2 point) :** On considère, sur une machine 16 bits (une adresse est encodée sur 16 bits), un espace mémoire physique de 1 Kilo-octets divisé en 4 pages de 256 Octets (pour rappel, $256 = 2^8$). On note ces pages P1, P2, P3, et P4. Le système d'exploitation implémente la pagination avec LRU. On souhaite exécuter un programme nécessitant 8 Ko, soit 32 pages logiques.

En supposant que les pages peuvent-être virtualisées sans limite d'espace de virtualisation, est-il possible d'exécuter un programme constitué de 32 pages logiques ? **Justifiez votre réponse en donnant le nombre maximum de pages logiques** que l'on peut utiliser pour un processus compte tenu de la configuration proposée.

## *Questions sur la synchronisation (4 points)*

Un programmeur a conçu une application de simulation relativement gourmande en puissance de calcul. Après analyse de son code, il se rend compte qu'il peut décomposer les calculs en 5 sous fonctions : f1, f2, f3, f4, f5. Ces fonctions n'utilisent que des variables locales où leurs paramètres d'entrée. On supposera que toutes ces fonctions prennent un ou plusieurs entiers en paramètre, et retournent un entier comme résultat (pour simplifier).

La logique d'exécution du programme qu'il a conçu est la suivante : au début de l'exécution 3 variables X Y et Z sont définies et servent à stocker les données d'entrée de la simulation. La simulation doit fournir les valeurs (R1 et R2) associées résultats de deux calculs : f4(f1(X), f2(Y)) et f5(f2(Y), f3(Z)). Une première implémentation de ce simulateur, **séquentielle**, repose sur le code suivant :

```
int main (){
  printf ("R1 vaut %d", f4(f1(X),f2(Y))) ;
  printf("R2 vaut %d", f5(f2(Y),f3(Z))) ;
  return 0 ;
}
```

Le programmeur pense qu'il serait utile de paralléliser les traitements pour accélérer ses simulations. Il propose de répartir les traitements dans 3 processus **partageant les variables Aux1 et Aux2**. Le code de ces trois processus est donné ci-après. Dans ce code, on fera l'hypothèse que **les variables partagées Aux1 et Aux2 sont initialisée à 0** (avant leur utilisation dans le code ci-après). Le programmeur utilise aussi 3 variables locales (non

partagées) : R1, R2 et RESLOC. On supposera que **les variables X, Y, et Z sont correctement initialisées au démarrage de l'exécution des processus**.

Le programmeur propose le code suivant :

| Processus 1 | Processus 2 | Processus 3 |
|---|---|---|
| 1. void main(){<br>2. RESLOC=f1(X)<br>3. R1=f4(RESLOC,Aux1) ;<br>4. printf("R1 vaut :");<br>5. printf("%d\n",R1);<br>6. } | 7. void main(){<br>8. Aux1= f2(Y) ;<br>9. R2=f5(Aux1,Aux2) ;<br>10. printf("R2 vaut :") ;<br>11. printf("%d\n",R2) ;<br>12. } | 13. void main(){<br>14. Aux2=f3(Z) ;<br>15. } |

Le développeur lance les trois processus simultanément et constate que les calculs ne correspondent pas du tout à ce qui était obtenu dans la version séquentielle.

**Question 9 (1 point) :** Expliquez pourquoi cette mise en parallèle des calculs **peut aboutir à des valeurs fausses pour R1 et R2**. Illustrez en proposant un scénario d'exécution amenant à des calculs faux.

**Question 10 (1 point) :** Un de ses collègues lui dit qu'un premier problème vient du code utilisé pour l'affichage : ce collègue pense que même si par chance les valeurs de R1 et R2 sont correctes, **l'affichage peut aboutir à un résultat impossible à interpréter**. **Ce collègue a raison ; expliquez pourquoi.**

Il est possible de synchroniser ces processus **en utilisant des sémaphores** pour garantir que les calculs et l'affichage soient systématiquement corrects.

**Question 11 (2 points) :** Proposez des modifications du code des trois processus permettant d'assurer leur synchronisation, **en utilisant des sémaphores**. Cette synchronisation doit avoir pour objectif d'autoriser le maximum de parallélisme entre ces processus tant que cela ne remet pas en cause la **correction des calculs et de l'affichage**. Pour répondre à cette question, vous devez indiquer, hors du code des processus, le nom des sémaphores que vous utilisez, ainsi que la valeur initiale de leurs compteurs. Par exemple, *Init(s, 54)* permet de déclarer un sémaphore s et d'initialiser sont compteur à 54. Vous pouvez ensuite utiliser, pour compléter le code ci-dessus, les fonctions *P(s)* et *V(s)* qui réalisent les opérations de prise et restitution de « jeton » d'un sémaphore s.

***Annexe à rendre***
Nom, Prénom :

**Tableau fournit pour répondre à la question 6:** pour rappel, P1, P2, et P3 sont ici des processus.

| | | | | | | | | | | | | | | | | | | |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| P1 | | | | | | | | | | | | | | | | | | |
| P2 | | | | | | | | | | | | | | | | | | |
| P3 | | | | | | | | | | | | | | | | | | |