Exercice 1 (Localité)
On considère le programme en C suivant (avec numérotation des lignes) :

```
1 main()
2 {
3     int i,j,k;
4     int tab1[1000];
5     float u,v;
6     int tab2[1000];
7     char msg[100];
8     int tab3[1000];
9     k = 0;
10    for (i=0; i<1000; i++)
11          { k = k + tab1[i]; }
12    printf("resultat = %d\n",k);
13    u = 0;
14    for (j=0; j<1000; j++)
15          { u = u*tab3[j]; }
16 }
```

1°) Expliquer le principe de localité à l'aide de ce programme. Donner sa chaîne des références en supposant que le compilateur alloue la mémoire donnée de manière contiguë dans l'ordre des déclarations, avec une taille de bloc de 500 sizeof(int) et en prenant sizeof(int) = 2 sizeof(char) et sizeof(float) = 2 sizeof(int) ; indiquer les phases du programme.

Correction :
On a 7 blocs en tout et 4 phases quelque soit l'exécution (ce qui n'est pas vrai en général).

i, j, k, tab1	*tab1*	*tab1, u, v, tab2*	*tab2*	*tab2, msg, tab3*	*tab3*	*tab3*

Phases : l9, l10/11, l12/13, l14/15

2°) On suppose dans la suite que la mémoire est paginée, que la taille des pages est de 500 sizeof(int) et que ce programme est le seul programme utilisateur en exécution. On ne considère que la mémoire réservée aux données et on néglige les autres besoins de mémoire utilisateur (code, etc.).

Quel est le nombre minimum MIN de cases de la mémoire physique des programmes utilisateurs à partir duquel il n'y aura pas de défaut de page à *chaque* exécution du corps de la deuxième boucle *for* ?

Correction :
MIN=3 : avec 2 cases, il faut à chaque fois charger la page de j puis celle de u tout en disposant de la page d'une partie de tab3.
Remarque : tab3 est sur trois pages donc la boucle comporte 3 "micro-phases" correspondant à la présence de la page du début, du milieu ou de la fin de tab3.

3°) L'algorithme de remplacement de pages utilisé est "Last Recently Used". On suppose que la mémoire physique comporte MIN cases. Donner la suite des pages présentes en mémoire physique en fonction de l'instruction en cours.

Exercice 2 (Phénomène de Belady)

On considère un système à mémoire paginée. L'algorithme de remplacement de page « Première entrée première sortie » ("FIFO") est le suivant. On maintient la liste des pages en mémoire physique dans l'ordre d'attribution des cases : la plus ancienne en tête. Lorsqu'il n'y a plus de case libre pour une nouvelle attribution, on place la plus ancienne page (celle qui est en tête) en zone de "swap" et l'on amène la page demandée en mémoire et en queue de liste.

1°) Soit un programme qui utilise 5 pages dans l'ordre suivant : 0 1 2 3 0 1 4 0 1 2 3 4. On suppose que l'on dispose de 3 cases de mémoire physique. Décrire la suite des fautes de pages.

Solution :

référence page		0	1	2	3	0	1	4	0	1	2	3	4
Plus jeune page		0	1	2	3	0	1	4	4	4	2	3	3
			0	1	2	3	0	1	1	1	4	2	2
Plus ancienne				0	1	2	3	0	0	0	1	4	4
Faute de page		x	x	x	x	x	x	x			x	x	

On a 9 fautes de pages.

2°) On dispose maintenant de 4 cases. Peut-on espérer plus ou moins de fautes de pages pour le même programme ? Décrire la suite des fautes de pages.

Solution :

référence page		0	1	2	3	0	1	4	0	1	2	3	4
Plus jeune page		0	1	2	3	3	3	4	0	1	2	3	4
			0	1	2	2	2	3	4	0	1	2	3
				0	1	1	1	2	3	4	0	1	2
Plus ancienne					0	0	0	1	2	3	4	0	1
Faute de page		x	x	x	x			x	x	x	x	x	x

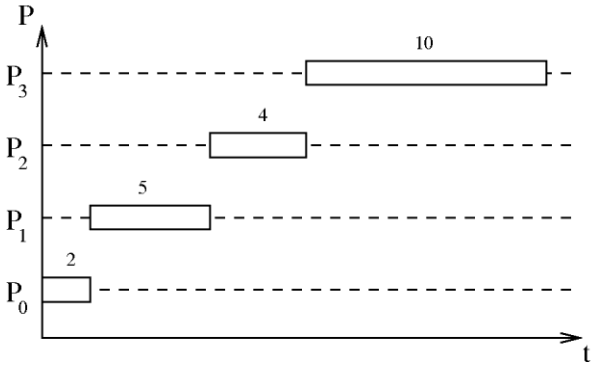
Plus de mémoire physique => moins de fautes de pages ?
On en a 10 avec 4 cases au lieu de 9 avec 3 cases!!!

Exercice 3 (Ordonnancement)

Nous considérons 4 jobs qui ont respectivement un temps d'exécution de 2, 5, 4 et 10 secondes et une priorité de 2, 4, 5 et 1 (la priorité 5 étant la plus élevée). Pour chacun des algorithmes d'ordonnancement suivants, donnez le temps moyen d'attente et la durée moyenne d'exécution.

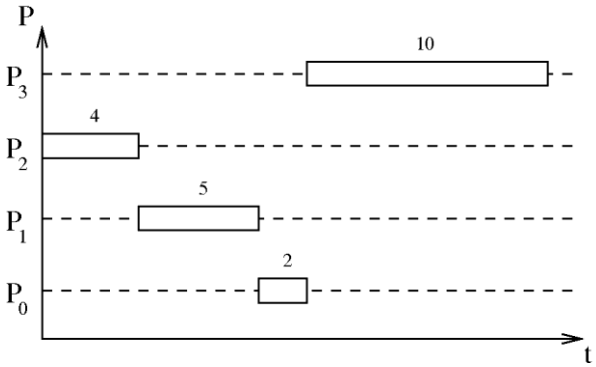
FIFO (exécutez dans l'ordre 2, 5, 4 et 10)

Temps moyen d'attente : (0 + 2 + 7 + 11)/4 = 5
Temps moyen d'exécution : [(0+2)+(2+5)+(7+4)+(11+10)]/4 = 10.25



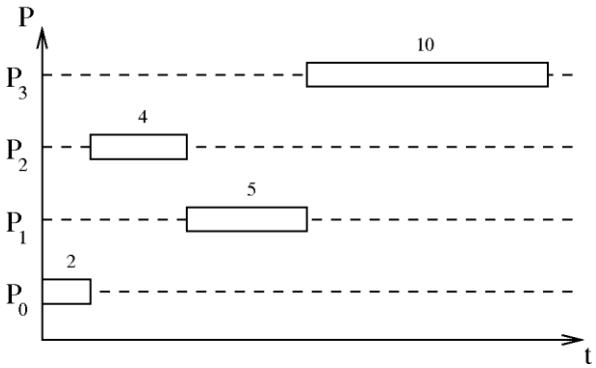
Ordonnancement par priorité

Temps moyen d'attente : (9 + 4 + 0 + 11)/4 = 6
Temps moyen d'exécution : [(9+2)+(4+5)+(0+4)+(11+10)]/4 = 11.25



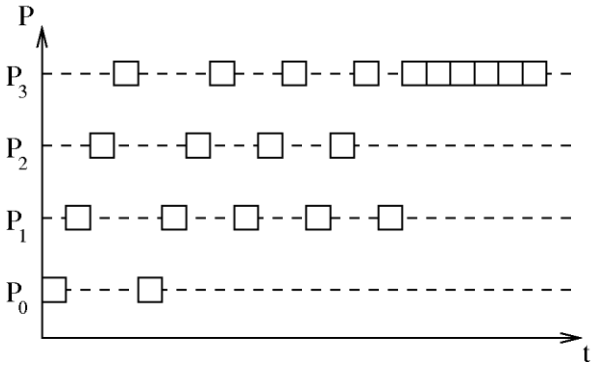
Job le plus court en premier

Temps moyen d'attente : (0 + 6 + 2 + 11)/4 = 4.75
Temps moyen d'exécution : [(0+2)+(6+5)+(4+2)+(11+10)]/4 = 10



Round Robin (Quantum=1)

Temps moyen d'attente : (0 + 1 + 2 + 3)/4 = 1.5
Temps moyen d'exécution : [(0+5)+(1+14)+(2+11)+(3+18)]/4 = 13.5


P
P3
P2
P1
P0
t