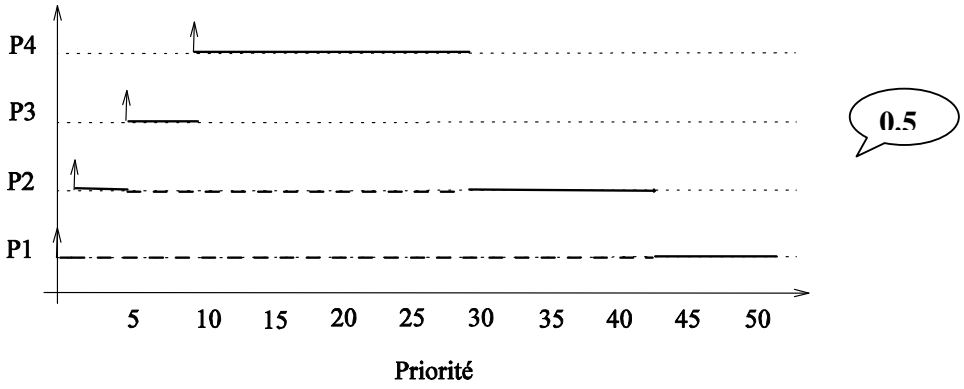
Corrigé Examen SE

Exercice1:(/4)

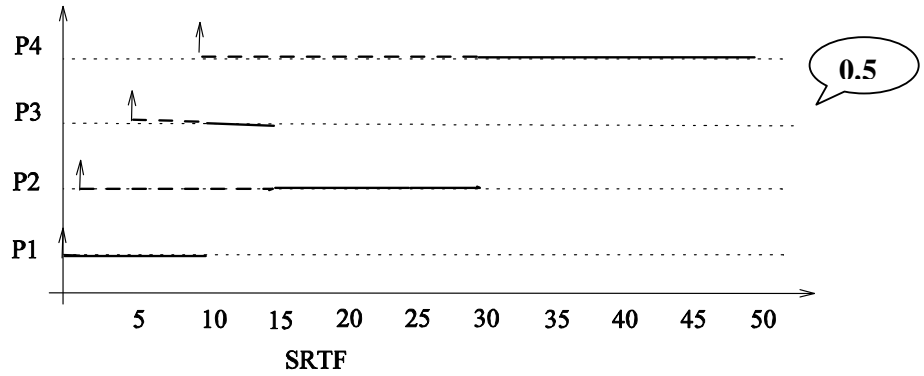
1) Le contexte d'un processus est tout ce qui définit son déroulement : code, données, registres, le compteur ordinal, et sa pile (**0.5**)

La commutation de contexte est quand un processus est interrompu pour être remplacé par un autre processus. Dans ce cas, le contexte du proc interrompu sera empilé dans la pile du système et le contexte du processus élu sera chargé dans le processeur. (**0.5**)

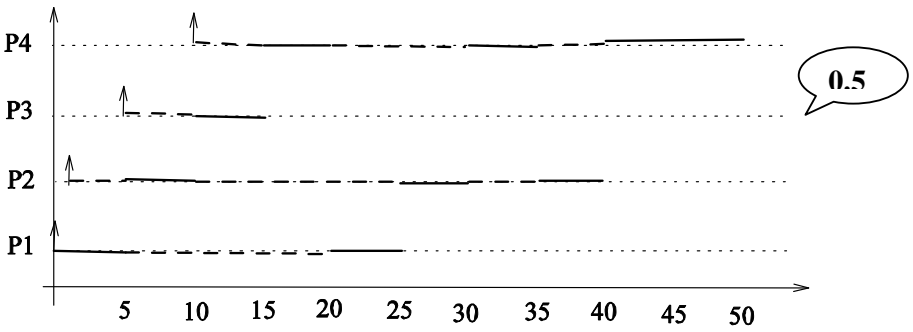
2) Ordonnancement



Priorité



SRTF



Tourniquet

T-att moyen (par priorité) : 32.5 **(0.5)**
T-att moyen (SRTF) :9,75 **(0.5)**
T-att moyen (RR) :16 **(0.5)**

SRTF (comme SJF pour le non-préemptif) donne de meilleurs résultats !

Processus	date d'arrivée	T-exec	priorité	T-att (par priorité)	T-att (SRTF)	T-att (RR)
P1	0	10	4	40	0	15
P2	1	15	3	25	14	24
P3	5	5	2	0	5	5
P4	10	20	1	0	20	20

Exercice2 :(/12)A/

1) l'algorithme parallèl (**1**)

```
begin
  T1 ;
  parbegin
     T4 ;
     begin
          parbegin
             T2 ;
              T3 ;
          parend
          T5 ;
     end
  parend
  T6 ;
end
```

2) Non, T3||T2 et T2||T4 mais T3→T4 ! la relation n'est pas transitive donc la relation de précédence ici n'est pas respectée et impossible de la représenter avec parbegin/parend (**0.5**)

3) Shared sempahore S1=1, S2=0, S3=0, S4=0, S5=0, S6=0 ; (**1**)

ProcA(){	ProcB(){	ProcC(){	ProcD(){	ProcE(){	ProcF(){
P(S1);					
T1;	P(S2);	P(S3);	P(S4);	P(S5);	P(S6);
V(S2);	T2;	T3;	P(S4);	P(S5);	P(S6);
V(S3);	V(S5);	V(S4);	T4;	T5;	T6;
V(S4);		V(S5);	V(S6);	V(S6);	
}	}	}	}	}	}
(0.25)	**(0.25)**	**(0.25)**	**(0.25)**	**(0.25)**	**(0.25)**

4) les sémaphores sont un outil puissant et robuste!

B) Soit le modèle lecteur rédacteur:

1) Définir le modèle et ses contraintes (voir cours) (**0.5**)

2 processus de type différent qui accèdent à une même structure de données. Les lecteurs la consultent (donc non-critique) et les rédacteurs la modifient (donc critique).

La lecture est simultanée, les lecteurs entre eux sont indépendants !

L'écriture est en exclusion mutuelle. Les rédacteurs entre eux sont en compétition en bloquant les lecteurs !

La variante choisie est implémentée ; il existe 4 variantes ! FIFO, etc.

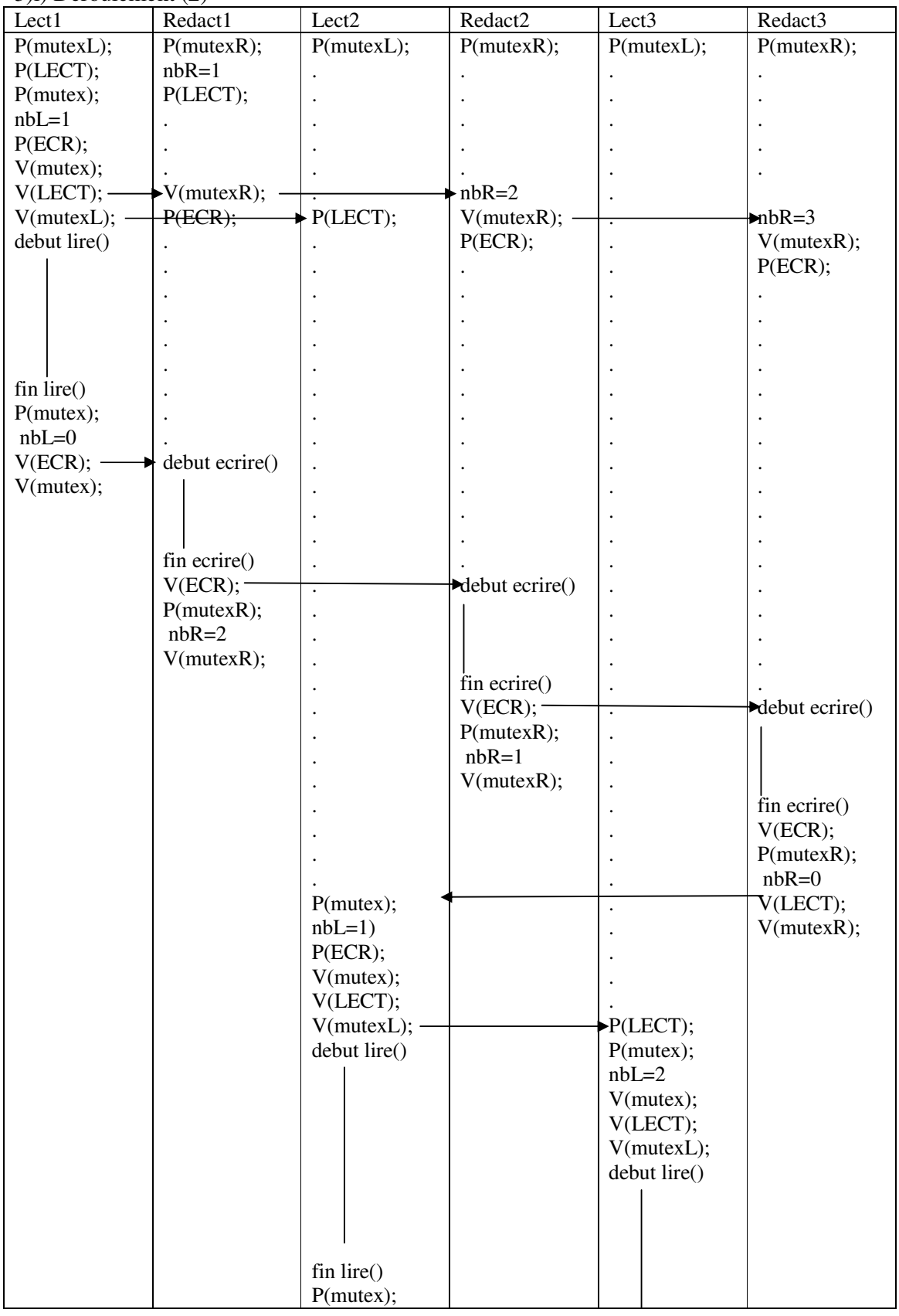
2) FIFO : Lect1 ;Redact1 ; Lect2 ; Redcat2 ; Lect3 ; Redact3 ;

Priorité aux lecteurs : Lect1||Lect2||Lect3 ; Redact1 ;Redact2 ;Redact3 ; (**0.5**)

Priorité aux rédacteurs : Lect1 ; Redact1; Redact2 ;Redact3; Lect2||Lect3 ; (**0.5**)

Priorité aux lecteurs sans famine des rédacteurs : Lect1 ;Redact1 ; Lect2||Lect3 ; Redact2 ;Redact3 ; (**0.5**)

3)i) Déroulement (**2**)



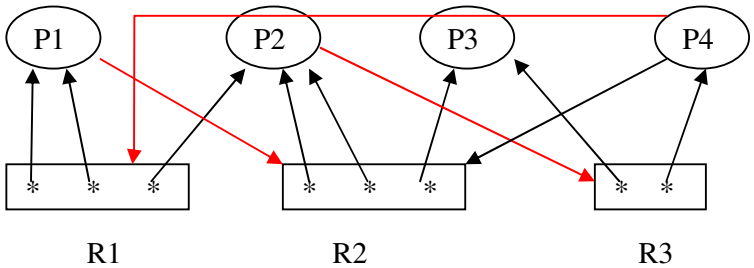
		nbL=1 V(mutex);		fin lire() P(mutex); nbL=0 V(ECR); V(mutex);	

i) Lect1 ; Redact1; Redact2 ;Redact3; Lect2| |Lect3 ;(**0.5**)
Priorité aux rédacteurs, car un rédacteur qui termine réveille d'abord un rédacteur (V(ECR))
Ce n'est pas FIFO car il n'ya pas de file d'attente commune pour indiquer l'ordre d'arrivée
Ce n'est pas la priorité aux lecteurs car un lecteur arrivant après un rédacteur sera bloqué par celui-ci.
Ce n'est pas la priorité aux lecteurs sans famine des rédacteurs car un rédacteur qui termine réveille d'abord un rédacteur qui est bloqué dans ECR !

ii) les contraintes du modèle sont bien vérifiées sur le déroulement !.
L'écriture est en EM : grâce à ECR qui protège l'écriture entre rédacteurs et lecteurs. Si une écriture est en cours alors les rédacteurs seront bloqués par ECR et le 1er lecteur sera bloqué par LECT et les autres seront bloqués dans mutexL. (**1**)
La lecture est simultanée : grâce à ECR qui empêche un rédacteur de passer. Si une lecture est en cours, alors tous les rédacteurs seront bloqués par ECR ! et les lecteurs arrivant après les rédacteurs bloqués seront bloqués par mutexL. (**1**)
mutexR : protège la variable nbR (nombre de rédacteurs) qui est une section critique entre les rédacteurs (**0.5**)
mutex: protège la variable nbLect (nombre de lecteurs) qui est une section critique entre lecteurs. (**0.5**)
mutexL : pour bloquer les lecteurs lors d'une écriture ou si un rédacteur est bloqué. (**0.5**)

Exercice3:(/4)
1) La différence entre la prévention statique, la prévention dynamique et la détection de l'interblocage? (**1**)
La prévention statique permet grâce à des règles fixes d'éviter tout interblocage (à priori)
La prévention dynamique permet grâce à l'analyse de chaque requête par l'algorithme du Banquier d'éviter tout interblocage (durant le déroulement, en temps réel)
La détection permet de vérifier si un système est en interblocage (à posteriori) pour le corriger !

2) Graphe (**1**)



etape1 : P3 ne demande aucune ressource, effacer ses allocations (R1=0 R2=1 et R3=1) (**0.25**)
etape2 : Requête de P1 sera satisfaite, effacer ses allocations (R1=2, R2=1, R3=1) (**0.25**)
etape3 : Requête de P2 sera satisfaite, effacer ses allocations (R1=3, R2=3, R3=1) (**0.25**)
etape4 : La requête de P4 sera satisfaite effacer ses allocations (R1=3, R2=3, R3=2) (**0.25**)
Le graphe est vide : système n'est pas en interblocage.

3) Si P3 avait une requête alors aucun processus ne peut être satisfait au départ (etape1 impossible) et donc il y'aurait interblocage totale. (**0.5**)
Correction de l'interblocage : Tuer P3 qui libérerait les autres processus et c'est sa requête qui a provoqué l'interblocage (**0.5**)

-FIN-