

Operating Systems: Training session

Raymond Namyst

Dept. of Computer Science

University of Bordeaux, France

<https://gforgeron.gitlab.io/se/>

Question de cours

www.wooclap.com/SEFOREVER

Pourquoi un processus ne peut pas monopoliser infiniment le processeur ?



- 1 Parce que le processeur n'a pas une durée de vie infinie de toute façon
- 2 Parce qu'à chaque interruption du temporisateur, le noyau reprend la main et peut effectuer un changement de contexte vers un autre processus
- 3 Parce que le noyau a le pouvoir de stopper à tout instant la séquence d'instructions exécutées le processeur
- 4 C'était une question-piège : si le processus appartient à root, il peut monopoliser le processeur..

wooclap



89 / 176



Question de cours

www.wooclap.com/SEFOREVER

Quelles sont les limites de l'option nachos -rs ?



- 1 Dans un vrai système, on ne peut pas passer d'option au démarrage
- 2 Dans un vrai système, les interruptions peuvent surgir n'importe quand, ce qui n'est pas le cas avec nachos
- 3 La graine du générateur pseudo-aléatoire rend les interruptions beaucoup trop prévisibles pour un programmeur aguerri

wooclap



95 / 176



Tram et voitures

```
While (1) {  
    sleep (PERIODE_SANS_TRAM);  
    rouge_clignotant = TRUE;  
    while (voitures_engagées > 0)  
        /* rien */ ;  
    sleep (TEMPS_PASSAGE_DU_TRAM);  
    rouge_clignotant = FALSE;  
}
```

```
While (rouge_clignotant)  
    /* rien */ ;  
    voitures_engagées++;  
    sleep(TEMPS_TRAVERSEE);  
    voitures_engagées--;
```

```
boolean rouge_clignotant = FALSE;  
int voitures_engagées = 0;
```

Tram et voitures

```
While (1) {  
    sleep (PERIODE_SANS_TRAM);  
    mutex_lock (&m);  
    rouge_clignotant = TRUE;  
    while (voitures_engagées > 0)  
        cond_wait (&wait_tram, &m) ;  
    mutex_unlock(&m);  
    sleep (TEMPS_PASSAGE_DU_TRAM);  
    mutex_lock (&m);  
    rouge_clignotant = FALSE;  
    cond_bcast (&wait_car);  
    mutex_unlock(&m);  
}
```

```
boolean rouge_clignotant = FALSE;  
int voitures_engagées = 0;  
mutex_t m;  
cond_t wait_tram, wait_car;
```

```
mutex_lock (&m);  
While (rouge_clignotant)  
    cond_wait (&wait_car, &m) ;  
voitures_engagées++;  
mutex_unlock(&m);  
sleep(TEMPS_TRAVERSEE);  
mutex_lock (&m);  
voitures_engagées--;  
cond_signal (&wait_tram);  
mutex_unlock(&m);
```

Histoires de groupes

```
thread_func (grp);  
  
P(mutex[grp]);  
nb[grp]++;  
if (nb[grp] == 1)  
    P(jeton);  
V(mutex[grp]);  
  
do_compute(grp);  
  
P(mutex[grp]);  
nb[grp]--;  
if (nb[grp] == 0)  
    V(jeton);  
V(mutex[grp]);
```

```
unsigned nb[N] = { 0, ... , 0 };  
semaphore mutex[N](1), jeton(1);
```

Reminder: Reader/Writer problem

```
Semaphore writeToken(1);  
int nbr = 0;  
Semaphore mutexR(1);  
Semaphore waitingRoom(1);
```

```
P(waitingRoom);  
P(mutexR);  
if (++nbr == 1) // pathfinder  
    P(writeToken);  
V(mutexR);  
V(waitingRoom);  
  
    read();  
  
P(mutexR);  
if (--nbr == 0) // last to leave  
    V(writeToken);  
V(mutexR);
```

```
P(waitingRoom);  
P(writeToken);  
V(waitingRoom);  
  
    write();  
  
V(writeToken);
```

Histoires de groupes

```
thread_func (grp);  
  
P(sas);  
P(mutex[grp]);  
nb[grp]++;  
if (nb[grp] == 1)  
    P(jeton);  
V(mutex[grp]);  
V(sas);  
  
do_compute(grp);  
  
P(mutex[grp]);  
nb[grp]--;  
if (nb[grp] == 0)  
    V(jeton);  
V(mutex[grp]);
```

```
unsigned nb[N] = { 0, ... , 0 };  
semaphore mutex[N](1), jeton(1);  
semaphore sas(1);
```

Histoires de groupes

```
thread_func (grp);  
  
    P(sas);  
    P(mutex[grp]);  
    nb[grp]++;  
    if (nb[grp] == 1)  
        P(jeton);  
    V(mutex[grp]);  
    V(sas);  
  
    P(limite);  
    do_compute(grp);  
    V(limite);  
  
    P(mutex[grp]);  
    nb[grp]--;  
    if (nb[grp] == 0)  
        V(jeton);  
    V(mutex[grp]);
```

```
unsigned nb[N] = { 0, ... , 0 };  
semaphore mutex[N](1), jeton(1);  
semaphore sas(1);  
semaphore limite(MAX);
```

Additional resources
available on

<http://gforgeron.gitlab.io/se/>