université de BORDEAUX

Collège Sciences et technologies

Année	2019-2020	**Type**	Devoir Surveillé
Master	Informatique		
Code UE	4TIN705U	**Épreuve**	Systèmes d'Exploitation
Date	6/11/2019	**Documents**	Non autorisés
Début	10h30	**Durée**	1h30

Correction du DS

1 Questions de cours (échauffement)

Question 1

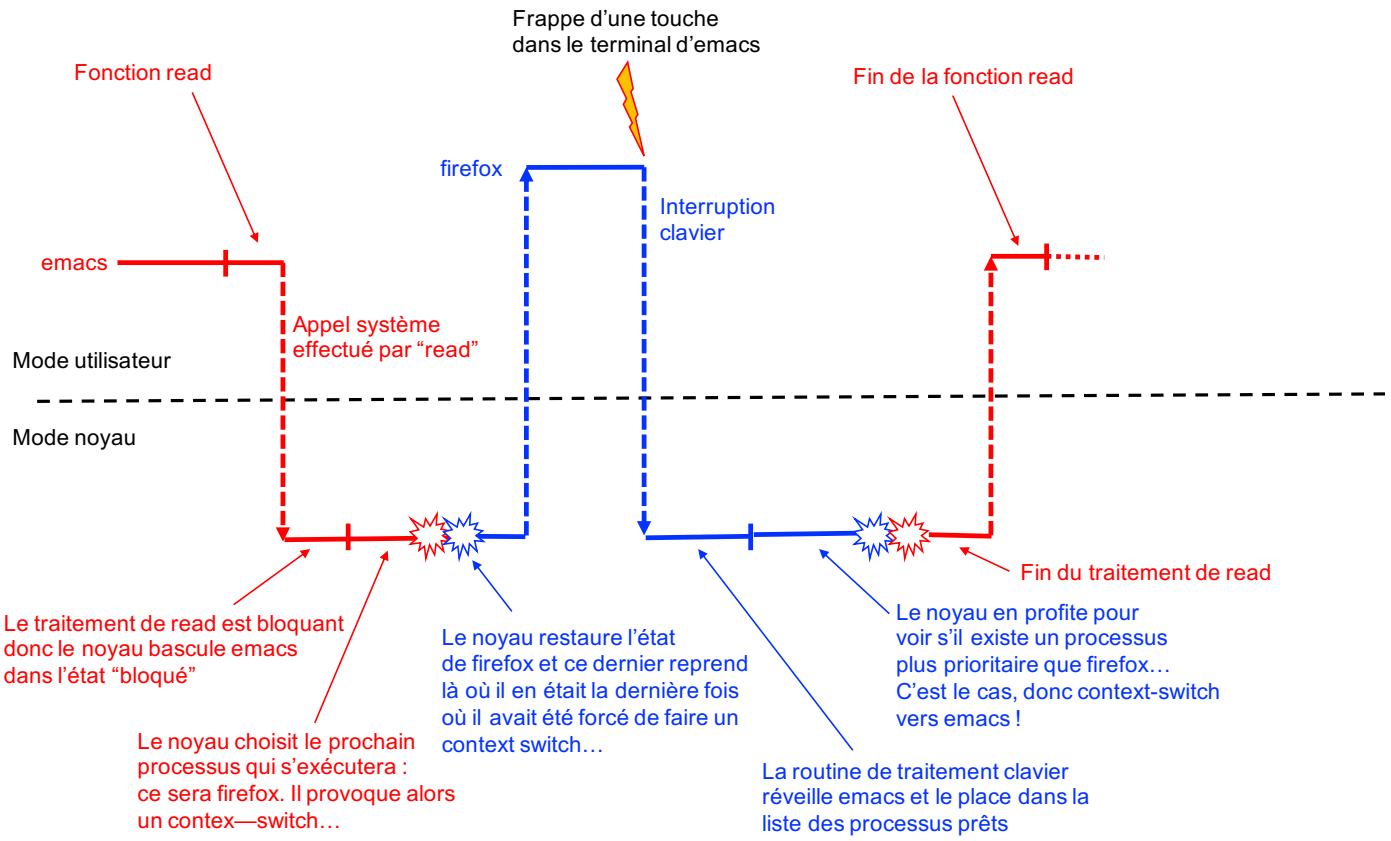

FIGURE 1 – Joli chronogramme décrivant une séquence possible d'évènements entre le début d'un appel système bloquant et sa terminaison.

Question 2 Le *noyau* est la partie minimale du système d'exploitation qui réside en mémoire RAM, et qui arbitre l'accès au matériel tout en fournissant aux processus une interface (i.e. les appels systèmes) permettant la gestion de processus, l'allocation de mémoire, les entrées/sorties, etc. Le noyau s'exécute une première fois lors du démarrage, puis à chaque fois qu'une interruption survient.

Les processus s'exécutent en mode utilisateur, c'est-à-dire qu'ils n'ont accès qu'à un sous-ensemble du jeu d'instruction du processeur, et ne peuvent manipuler que des adresses mémoire virtuelles. La seule possibilité pour un processus d'effectuer une écriture dans un fichier disque, par exemple, est de déclencher volontairement une interruption pour basculer dans le noyau et exécuter un code entièrement contrôlé par le noyau. Une tentative d'accès direct au contrôleur disque serait soldée par une interruption de type *Illegal Instruction*.

Question 3 Deux séquences possibles :

$P_3(3) \xrightarrow{quantum} P_3(2) \xrightarrow{quantum} P_3(1)P_2(2) \xrightarrow{quantum} P_2(1) \xrightarrow{quantum} P_2(0)P_3(1) \xrightarrow{quantum} P_3(0)P_1(1) \xrightarrow{quantum} P_1(0)$

ou

$P_3(3) \xrightarrow{quantum} P_3(2) \xrightarrow{quantum} P_3(1)P_2(2) \xrightarrow{quantum} P_2(1) \xrightarrow{quantum} P_2(0)P_1(1) \xrightarrow{quantum} P_1(0)P_3(1) \xrightarrow{quantum} P_3(0)$

2 Le grand bain

Question 1 Il est facile de voir que deux threads peuvent parcourir les casiers simultanément, choisir le même casier `i`, et le marquer à 1 tous les deux sans s'en apercevoir...

Question 2 Trouver un casier...

```
bool casiers [MAX_CASIERS] = { FAUX, ..., FAUX }; // FAUX = LIBRE

unsigned libres = MAX_CASIERS;
mutex_t mutex;
cond_t attente_casier;

// retourne le numéro du casier obtenu (sorte de "bracelet" que le baigneur portera au poignet)
unsigned trouver_casier ()
{
  mutex_lock (&mutex);

  while (libres == 0)
    cond_wait (&attente_casier, &mutex);

  libres--;

  // Plus besoin du while(1)
    for (int i = 0; i < MAX_CASIERS; i++)
      if (casiers [i] == FAUX) {
        casiers [i] = VRAI;
        mutex_unlock (&mutex);
        return i;
      }
    assert (FALSE); // On ne devrait jamais arriver ici !
}

void liberer_casier (int num)
{
  mutex_lock (&mutex);
  casiers [num] = FAUX;
  libres++;
  cond_signal (&attente_casier);
  mutex_unlock (&mutex);
}
```

Question 3 Avec un seul guichet...

```
int reste = CAPACITE_MAX; // nombre de places disponibles à la vente
mutex_t mutex_file;
cond_t attente_guichet;
unsigned guichet_libre = 1;

int payer_au_guichet ()
{
  mutex_lock (&mutex_file);

  if (reste == 0) { // on voit tout de suite l'écriteau "complet"
    mutex_unlock (&mutex_file);
    return -1;
  }

```

```
  reste--;

  while (guichet_libre == 0)
    cond_wait (&attente_guichet, &mutex_file);

  guichet_libre--;

  mutex_unlock (&mutex_file);

  sleep (DELAI_PAIEMENT);

  mutex_lock (&mutex_file);
  guichet_libre++;
  cond_signal (&attente_guichet);
  mutex_unlock (&mutex_file);

  return 0; // tout va bien
}
```

Question 4 La file devient dissuasive à partir de `MAX_FILE`

```
int reste = CAPACITE_MAX; // nombre de places disponibles à la vente
mutex_t mutex_file;
cond_t attente_guichet;
unsigned guichet_libre = 1;
unsigned longueur_file = 0;

int payer_au_guichet ()
{
  mutex_lock (&mutex_file);

  if (reste == 0 || longueur > MAX_FILE) {
    mutex_unlock (&mutex_file);
    return -1;
  }

  reste--;
  longueur_file++;

  while (guichet_libre == 0)
    cond_wait (&attente_guichet, &mutex_file);

  guichet_libre--;
  longueur_file--; // on ne compte pas les personnes qui sont en train
                   // de payer

  mutex_unlock (&mutex_file);

  ... // la suite ne change pas
}
```

Question 5 Avec plusieurs guichet...

```
...
unsigned guichet_libre = NB_GUICHETS;
...

// Le reste ne change pas !
```