Programmation système – TP2

Objectifs du TP :

1. character device
2. kernel_thread
3. File d'attente

1 Character device

Prise en compte

```
#include <linux/fs.h>
int register_chrdev(unsigned int major, char * name,
                    struct file_operations fops);
```

major : identifiant du pilote de périphéique dans le noyau (199 pour le TP)

name : nom du pilote

fops : structure de pointeurs de fonctions

retour < 0 : erreur

Déclaration et initialisation de la stucture *file_operations* **:**

```
static int maFonctionOpen(struct inode * inode,
   struct file  * file) {
  ...
  return 0;
}

static struct file_operations fops = {
  .open = maFonctionOpen,
};
```

Vérification

```
# less /proc/devices
```

Suppression

```
void unregister_chrdev(unsigned int major, char * name);
```

Liaison avec un nom de fichier spécial

```
mknod /dev/monPeripherique c $(MAJOR) $(Minor)
chmod 644 /dev/monPeripherique
```

Pour provoquer l'ouverture

```
cat > /dev/monPeripherique
^C
```

2 File d'attente

Déclaration

```
#include <linux/sched.h>
static wait_queue_head_t wq;
```

Initialisation (file vide)

```
init_waitqueue_head(&wq);
```

Mise en attente

```
/* macro */
wait_event(wq, condition);
```

Réveil de tous les threads dans la file

```
wake_up(&wq);
```

3 Kernel_thread

Création

```
int kernel_thread(int(*f)(void *), void * arg, 0);
```

Terminaison et synchronisation

```
static struct completion comp;

/* à l'initialisation du module */
...
init_completion(&comp);
...

/* dans la fonction du thread */
int thread_function(void * dummy) {
  ...
  complete_and_exit(&comp, 0);
}

/* a priori dans clean_up (déchargement du module) pour ce tp */
void clean_up(void) {
  ...
  wait_for_completion(&comp);
  ....
}
```