Cognoms: _____ Nom: _____ DNI: _____ Núm Disc:_____

CONCEPTES AVANÇATS DE SISTEMES OPERATIUS (CASO)

Facultat d'Informàtica de Barcelona, Dept. d'Arquitectura de Computadors, curs 2012/2013 – 1Q

Examen de Laboratori 21/12/12

L'examen és individual
Responeu en l'espai assignat
Indiqueu COGNOMS, NOM i DNI (per aquest ordre)
Podeu consultar llibres i apunts, però no es poden usar ordinadors portàtils o mòbils
Justifiqueu totes les respostes
Temps: 2 hores

1. Introducció

Tenim un ordinador amb Ubuntu Linux i volem aprendre a accedir a la informació interna del sistema operatiu Linux, a través d'un dispositiu de caràcter. Aquesta és la informació a la que accedirem:

<linux/sched.h>:

```
struct task_struct {
        volatile long state;    /* -1 unrunnable, 0 runnable, >0 stopped */
    ...
        int on_cpu;
    ...
        pid_t pid;
    ...
        cputime_t utime, stime, utimescaled, stimescaled;
    ...
};
```

Són diversos camps de l'estructura dels processos. El significat de cada camp és el següent:

- state: l'estat del procés
- on_cpu (booleà): si està en un processador (1) o no (0)
- pid: el pid del procés
- *time*: diversos temps del procés (usuari, sistema)

Per copiar aquesta informació al nivell d'usuari farem un **open** per obrir el dispositiu **/dev/pinfo** i un **read** del descriptor obtingut. Això ens donarà la informació del propi procés que està llegint el dispositiu. La crida corresponent al **read** omplirà una variable d'aquest tipus, per poder tornar la informació a l'usuari:

```
struct user_process_info {
   long      upi_state;
   int       upi_on_cpu;
   pid_t     upi_pid;
   cputime_t upi_utime, upi_stime, upi_utimescaled, upi_stimescaled;
};
```

Si volem aconseguir la informació d'un procés diferent de l'actual, farem servir la crida **ioctl** sobre el descriptor de fitxer obtingut en l'open. Al fer això, passarem com a paràmetre del ioctl el pid del procés que volem accedir a partir d'ara. El controlador de dispositiu comprovarà que el procés existeix, i deixarà la informació interna del dispositiu modificada per tornar la informació d'aquest procés en els següents lectures. La crida ioctl té aquesta interfície:

```
int ioctl (int fd, int request, long argument);
```

i l'aplicació la podrà fer servir així:

```
res = ioctl(fd, IOCTL_CHANGE_PROCESS, pid);
if (res < 0) { perror ("ioctl pinfo"); exit (1); }
```

2. El codi font proporcionat

Entreu al sistema:

username: caso

password: caso

Tenim aquest paquet amb el codi font parcial del controlador de dispositiu de caràcter: timedrv.tar.bz2. Es tracta d'un mòdul de Linux, que conté el controlador. Hi ha un altre paquet amb un directory amb els binaris (**binary**) per què pogueu provar el funcionament esperat.

Creeu un directori de treball (per exemple "work") i descomprimiu i extraieu el contingut d'aquest fitxer a dins del directori de treball.

Estudieu els fitxers que conté el paquet:

- Makefile // per compilar-ho tot
- user_process_info.h // definicions d'estructures i macros
- interface.c // codi del controlador de dispositiu de caràcter
- implementation.c // no usat (i no cal que l'useu)
- mytimes.c // aplicació simple que llegeix la informació pròpia i l'escriu per pantalla
- yourtimes.c // aplicació que llegeix la informació d'altres processos

3. 1a fase: lectura de la informació

En aquest apartat heu d'implementar la funció:

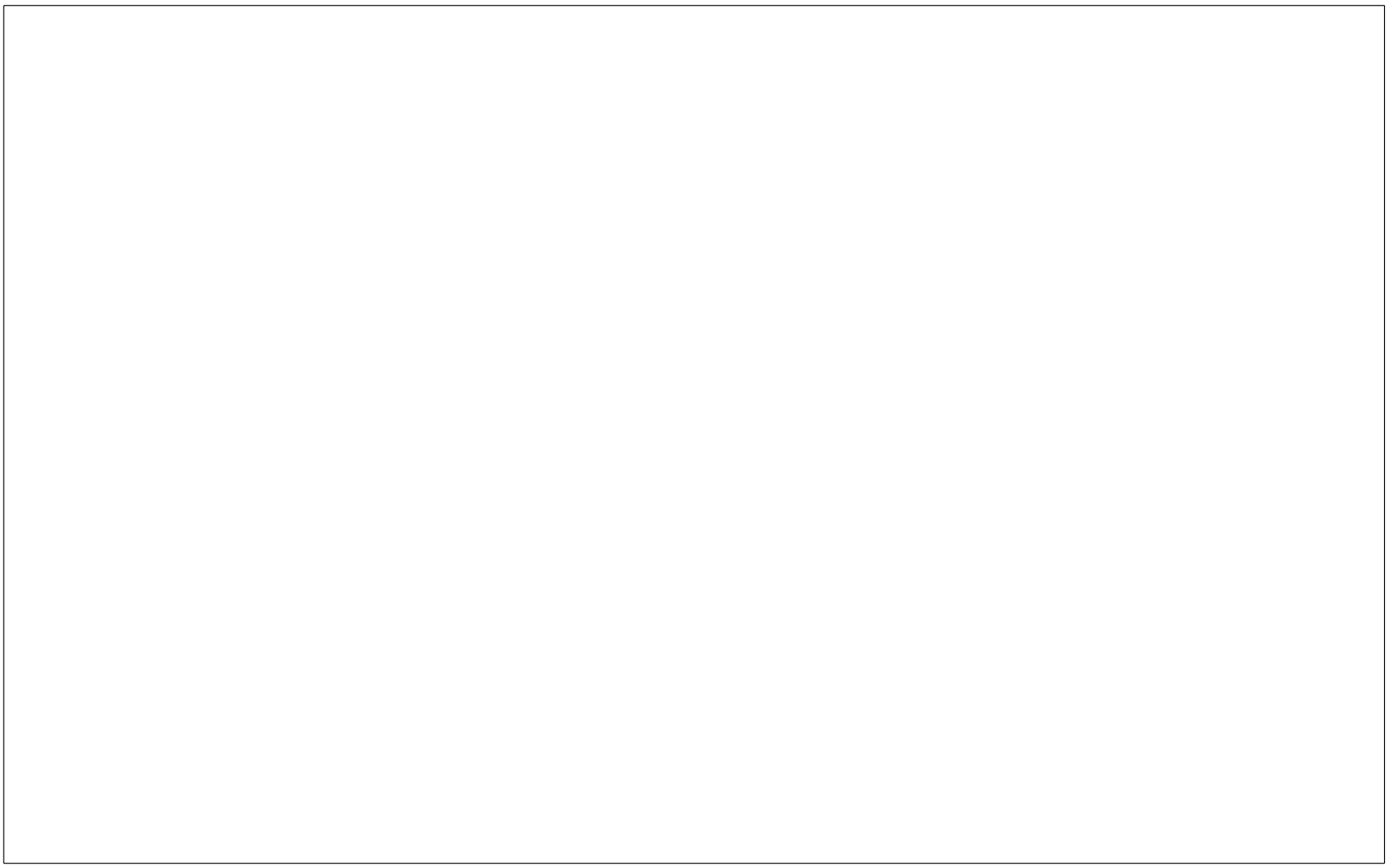
```
ssize_t  mychardrv_read (
                struct file * f,
                char __user * address,
                size_t size,
                loff_t * offset);
```

per tal que realitzi els següents passos:

- comprovar que la mida que demana l'usuari almenys es tan gran com la mida de l'estructura **user_process_info**. Altrament, retornar l'error -ENOBUFS.
- usar la funció **int get_process (int pid, struct task_struct ** p);**, que busca el procés indicat pel **pid** donat com a paràmetre (i que podeu treure de la variable **current_pid** del controlador de dispositiu). Aquesta funció, si no troba el procés, retorna un resultat d'error (< 0), que vosaltres podeu retornar directament a l'usuari. Si troba el procés, retorna zero (0) i canvia el punter passat per referència (p), de forma que apunti al *task_struct* del procés **pid**.
- copiar la informació dels camps indicats a la introducció a l'estructura *user_process_info* (variable **internal_copy** de la mateixa funció). Feu les assignacions camp a camp, per què el codi resulti senzill.
- fa el **copy_to_user** per copiar les dades de l'estructura **internal_copy** a l'adreça proporcionada per l'usuari (**address**). La interfície de copy_to_user és:

  → `int copy_to_user (void __user * target_address, void * source_address, int num_bytes);`
- si copy_to_user retorna 0 és que tot ha anat bé i podem retornar la mida copiada a l'usuari, que serà el retorn de la crida a sistema **read**.
- si el copy_to_user retorna un número diferent de zero, retornarem -EFAULT a l'usuari.

Un cop aquesta funció està implementada, el mòdul/controlador està llest per ser introduït al sistema. Feu l'**insmod** i proveu l'aplicació **mytimes**. Quan veieu que mytimes no dóna errors, podeu escriure aquí la sortida que us dóna:



4. 2a fase: canvi del procés d'on llegir la informació

Programeu ara la funció **mychardrv_ioctl**. La seva interfície és:

```
long mychardrv_ioctl(struct file * f, unsigned int request, unsigned long pid);
```

i la seva funcionalitat és la següent:

- comprovar que el pid donat com a paràmetre és encara un procés vàlid. Per fer això usarà la funció ja coneguda **get_process**.
- si get_process retorna zero (0), indicant que la cerca del pid ha trobat el procés encara actiu, la funció farà un set_inspect_process(pid) i retornarà un zero (0).
- altrament, la funció retornarà l'error indicat per get_process.

Aquesta funció **mychardrv_ioctl** s'ha d'apuntar amb el camp **unlocked_ioctl** a l'estructura **file_operations** que tenim al mateix fitxer interface.c.

Compileu el mòdul correctament i inserteu-lo en el sistema. Un cop s'inserti amb èxit, podeu provar l'aplicació **yourtimes**:

$ yourtimes 1 2 3 $$ me $$

Escriviu aquí les 3 primeres línies que obteniu i aviseu a un professor per tal que verifiqui la sortida i signi a continuació:

Comentari (si s'escau) i signatura del professor:

5. Crítiqueu...

Executeu la següent combinació de comandes:

$ mytimes & mytimes

i estudieu la sortida que donen a l'executar-se en paral.lel.

Critiqueu aquí el funcionament del controlador de dispositiu en aquest cas i proposeu una solució (no cal que la implementeu):