Konrad Gotfryd

STER
Dokumentacja projektu.

Temat projektu: Sterownik do komunikacji IPC

Opis funkcji użytkownika (API):

int event_set(char *name) – funkcja dopuszczająca proces wywołujący tę funkcję do wysyłania zdarzeń o nazwie spod wskaźnika name. Funkcja zwraca 0 gdy proces został dopuszczony do wysyłania podanego zdarzenia, lub w przypadku błędu ustawia wartość -1 i ustawia odpowiednio wartość zmiennej errno.

int event_wait(char *name) – funkcja, której celem jest oczekiwanie na nazwane zdarzenie. Funkcja zwraca 0, gdy oczekiwanie się powiodło, lub wartość -1, gdy z jakiegoś powodu funkcja nie może oczekiwać, lub oczekiwanie zostało przerwane.

int event_wait_group(char **events, int events_cnt) – oczekiwanie na jedno z grupy zdarzeń. Parametr events_cnt określa liczbę zdarzeń, która ma być pobrana ze wskaźnika char **events. Funkcja zwraca 0, gdy oczekiwanie się powiodło, lub wartość -1 w przeciwnym wypadku. Stosownie do błędu ustawia errno.

int event_throw(char *name) – rzucanie zdarzeniem. Funkcja zwraca 0, gdy rzucenie zdarzeniem jest możliwe, -1 w przeciwnym wypadku. Errno ustawione stosownie do błędu.
int event_unset(char *name) – Usuwanie nazwanego zdarzenia ze sterownika. Nieusunięcie zdarzenia przez proces, który wcześniej ustawia zdarzenie przez event_set, powoduje nieprawidłową pracę sterownika (jest możliwość obejścia problemu na platformie ARM).

int event_check_error_exit(int rt, char *st) – funkcja sprawdzająca, czy wystąpił błąd. Jeśli tak, wyświetlany jest komunikat spod stringu *st i komunikat związany ze znaczeniem errno, następnie proces jest przerywany.

int event_check_error(int rt, char *st) – to samo co event_check_error_exit, lecz bez przerywania procesu.

Opis sterownika.

Najważniejsze struktury sterownika:

```
struct event {
     char *name;
     struct completion **wait;
     char **completed_by;
     unsigned int s_comp;
     unsigned int g_comp;
```

```
     struct list_head element;
     struct task_struct **proc_throws;
     struct task_struct **proc_waits;
};

struct events {
     const char *driver_name;
     struct cdev *cdev;
     dev_t dev;
     struct class *class;
     struct file_operations fops;
     struct device *device;
     struct list_head event_list;
     struct mutex lock;
     unsigned int event_cnt;
};
```

int events_set(struct events *cmc, const char __user *buf) – funkcja tworzy zdarzenie lub do już utworzonego zdarzenia wpisuje w tablicę proc_throws wskaźnik struktury task_struct procesu, który wywołał funkcję event_set z userspace.

int events_wait(struct events *cmc, const char __user *buf) - funkcja bada, czy dane zdarzenie o nazwie z buf istnieje, przez znalezienie zdarzenia o takiej nazwie w cmc->event_list. Jeśli nie ma, zwraca błąd -EINVAL. Jeśli jest, bada czy proces może oczekiwać na te zdarzenie (możliwość zajścia zagłodzenia). Jeśli nie istnieje taka możliwość, następuje uśpienie procesu w funkcji wait_for_completion_interruptible().

int events_group_wait(struct events *cmc, const char __user *user_buf) – funkcja podobna do events_wait, z tym, że zamiast oczekiwania na jedno zdarzenie, tworzona jest structura completion specjalna do tego oczekiwania i wskaźnik tej struktury jest wprowadzany do każdego zdarzenia, na które ma funkcja oczekiwać.

int events_throw(struct events *cmc, const char __user *buf) – funkcja sprawdza, czy podane zdarzenie istnieje. Jeśli istnieje, następuje wywołanie funkcji complete_all, która kończy oczekiwania.

int events_unset(struct events *cmc, const char __user *buf) – funkcja sprawdza, czy możliwe jest usunięcie zdarzenia, albo usunięcie procesu związanego ze zdarzeniem. Funkcja sprawdza, czy usunięcie danego zdarzenia nie spowoduje nieskończonego oczekiwania na zdarzenie innych procesów (istnieje możliwość, że proces, który ma zostać wyrejestrowany jest tym, przez który nie dochodzi do deadlocku).