Линукс курс 2016/2017

Спрега са улазним подсистемом

ЦИЉ

Омогућити приступ I^2C уређају из user space-а коришћењем улазног подсистема.

ИСХОД

Након ове вежбе ћете моћи да:

- Омогућите приступ догађајима user space-у кроз улазни подсистем (спрегу) коришћењем polling API језгра оперативног система Линукс за улазне уређаје (kernel space перспектива).
- Рукујете грешкама приликом регистрације или алокације на коректан начин.
- Користите улазну спрегу са више разумевања и више детаља (user space перспектива).

ПОСТАВКА

Позиционирајте први терминал на путању `~/linux-kernel-labs/src/linux` ради превођења језгра и DTB (останите на истом branch-у). Позиционирајте други терминал на путању `~/linux-kernel-labs/modules/nfsroot/nunchuk` ради превођења модула. Не заборавите да у оба терминала поставите променљиве `ARCH` и `CROSS_COMPILE` на одговарајуће вредности.

ДОДАВАЊЕ ПОДРШКЕ ЗА POLLING УЛАЗНИ УРЕЂАЈ У КЕРНЕЛ

Nunchuk нема прекиде (interrupt) којима би обавестио I^2C мастер да је његово стање промењено. Стога је једини начин за приступ подацима уређаја и

детектовање промена прозивање (polling) његових регистара коришћење улазног polling API описаног на предавањима.

Поново преведите кернел са статичком подршком за `polled input device` (`CONFIG_INPUT_POLLDEV=y`) и за `event interface` (`CONFIG_INPUT_EVDEV`). У подразумеваној конфигурацији, они су доступни као модули, што је мање погодно. Прекопирајте `zImage` датотеку и покрените поново плочу.

РЕГИСТРОВАЊЕ УЛАЗНЕ СПРЕГЕ

Прво што треба урадити је додавање улазног уређаја у систем. То је могуће урадити у следећим корацима:

- Декларисати показивач на `input_polled_dev` структуру у функцији `probe`. Можете га назвати `polled_input`. Није погодно користити глобалну променљиву јер ваш руковалац треба да има подршку за више уређаја прикључених истовремено.
- Алоцирајте поменуту структуру у истој функцији, користећи функцију `input_allocate_polled_device()`.
- Такође, дефинишите показивач на `input_dev` структуру. Можете га назвати `input`. Алокација није потребна јер је он већ део `input_polled_dev` структуре и истовремено је алоциран. Користићемо га само као пречицу да бисмо код одржали што једноставнијим.
- У функцији `probe()` додајте улазни уређај у систем позивом `input_register_polled_device()`.

У овој фази се најпре уверите да се ваш модул успешно преводи (по потреби додајте недостајућа заглавља).

Линукс курс 2016/2017

РУКОВАЊЕ ГРЕШКАМА У PROBE ФУНКЦИЈИ

У коду који сте написали, проверите да ли на одговарајући начин реагујете у случају могућих грешака:

- Пре свега, увек на одговарајућ начин проверите повратне вредности функција и евидентирајте узроке грешака.
- Ако је позив функције `input_register_polled_device()` неуспешан, морате ослободити `input_polled_dev` структуру пре него што вратите код грешке, односно пре него што се заврши извршавање функције. Ако то не урадите створићете цурење меморије (memory leaks) у кернелу. Уопштено говорећи, неуспешно отпуштање онога што је претходно било алоцирано или регистровано може довести до проблема односно може спречити поновно учитавање модула.

Да бисте ово коректно имплементирали без сувишног копирања кода или ружног и непрегледног кода, ово је место где се препоручује употреба `goto` наредби.

ИМПЛЕМЕНТАЦИЈА ФУНКЦИЈЕ REMOVE()

У овој функцији је потребно отпустити све ресурсе које је заузела или регистровала функција `probe()`.

Међутим, ово није тривијално као што је била имплементација функције `probe()`. Како наш руковалац треба да подржава више уређаја истовремено, одлучили смо се да не користимо гобалне променљиве, а као последица тога сада не можемо користити такве променљиве да бисмо ослободили одговарајуће ресурсе.

Ово доводи до веома важног аспекта модела уређаја: потребе да се показивачи одрже између физичких уређаја (уређаја којима рукује физичка магистрала, I^2C у нашем случају) и логичких уређаја (уређаја којима рукују подсистеми, као што је улазни подсистем у нашем случају).

На овај начин, када је позвана функција `remove()` (типично када магистрала детектује да је неки од уређаја уклоњен) можемо открити који је логички уређај потребно одјавити (unregister). Обрнуто, када имамо догађај на логичкој страни (као што је отварање или затварање улазног уређаја по први пут), можемо открити којем I^2C slave уређају тај догађај припада, како бисмо урадили специфичне кораке са физичким уређајем (hardware).

Ово је типично имплементирано кроз стварање структуре приватних података за управљање уређајем и имплементацију показивача између физичког и логичког света.

Додајте следеће дефиниције у ваш код:

```
struct nunchuk_dev {

struct input_polled_dev *polled_input;

struct i2c_client *i2c_client;

};
```

Затим у функцији `probe()`, декларишите инстанцу ове структуре:

```
struct nunchuk_dev *nunchuk;
```

Потом алоцирајте једну такву структуру за сваки нови уређај:

```
nunchuk = devm_kzalloc(&client->dev, sizeof(struct nunchuk_dev), GFP_KERNEL);

if (!nunchuk) {

dev_err(&client->dev, ”Failed to allocate memory\n”);

return -ENOMEM;

}
```

Приметите да до сада нисмо видели функције за алоцирање меморије у кернелу.

Све што за сада треба да знате је да се свака алокација или регистрација коју направи `devm_` функција припаја структури уређаја (device structure). Када се

уређај или цео модул уклоне, све такве алокације или регистрације се аутоматски пониште. Ово у значајној мери поједностављује код руковаоца.

Још увек нисмо објаснили ни функције `dev_*` за евидентирање порука (оне се у основи користе да би јасно истакле за који уређај је евидентирана порука везана).

Сада имплементирајте показиваче:

```
nunchuk->i2c_client = client;

nunchuk->polled_input = polled_input;

polled_input->private = nunchuk;

i2c_set_clientdata(client, nunchuk);

input = polled_input->input;

input->dev.parent = &client->dev;
```

Водите рачуна да овај код треба да стоји пре регистровања улазног уређаја. Не би требало омогућити рад уређаја са непотпуним информацијама, односно док све информације нису уписане (у супротном би могло доћи до трке).

Када је све ово урађено, коначно имамо све што је неопходно да би се имплементирала функција `remove()`. Потражите примере на слајдовима о I^2C и улазним уређајима или директно у Линукс кернел коду!

Преведите поново модул, учитајте га, па уклоните неколико пута, како бисте се уверили да је све успешно пријављено односно одјављено.

ДОДАВАЊЕ ИСПРАВНИХ ИНФОРМАЦИЈА О РЕГИСТРАЦИЈИ УЛАЗНОГ УРЕЂАЈА

Пре саме регистрације улазне структуре, морамо јој додати још информација. То је разлог због којег у овом тренутку још увек добијате упозорења приликом учитавања модула, односно пријаве новог уређаја:

```
input: Unspecified device as /devices/virtual/input/input0
```

Додајте линије кода које следе (такође пре регистрације, наравно):

```
input->name = "Wii Nunchuk";

input->id.bustype = BUS_I2C;

set_bit(EV_KEY, input->evbit);

set_bit(BTN_C, input->keybit);

set_bit(BTN_Z, input->keybit);
```

Поново преведите модул и покушајте поново да га учитате. Сада би требало да видите међу кернел порукама да је „Unspecified device type“ замењено са „Wii Nunchuck“ и да је физичка путања до уређаја такође пријављена.

ИМПЛЕМЕНТАЦИЈА POLLING РУТИНЕ

Остало је још да имплементирамо функцију која ће прозивати nunchuk регистре у предвиђеном временском интервалу.

Направите функцију `nunchuck_poll()` са одговарајућим прототипом (пронађите га по угледу на дефиницију `input_polled_dev` структуре).

Најпре додајте линије којима преузимате физички I^2C уређај из `input_polled_dev` структуре. Ту ће вам бити потребна приватна `nunchuk` структура.

Сада када имате показивач (handle) ка физичком I^2C уређају, код за читање nunchuk регистара можете пребацити у ову функцију. Можете уклонити двоструко читање стања уређаја јер ће polling функција свакако правити периодична читања[1].

На крају polling рутине, последње што треба урадити је постављање догађаја и обавештење језгра улазног подсистема. Под претпоставком да је `polled_input` назив `input_polled_dev` параметра ваше polling рутине:

[1]Приликом пребацивања кода, мораћете да рукујете комуникационим грешкама на мало другачији начин, с обзиром да функција `nunchuk_poll()` има `void` тип. Када је функција за читање регистара неуспешно завршена, можете користити `return;` исказ уместо исказа `return value;`

```
input_event(polled_input->input,
EV_KEY, BTN_Z, zpressed);
input_event(polled_input->input,
EV_KEY, BTN_C, cpressed);
input_sync(polled_input->input);
```

Вратите се на функцију `probe()`. Последње што требада урадите је да декларишете нову polling функцију (погледајте слајдове са предавања уколико сте заборавили детаље) и наведете polling интервал од 50 ms.

Проверите да ли се код преводи и учитава.

ТЕСТИРАЊЕ УЛАЗНЕ СПРЕГЕ

Тестирање улазног уређаја је једноставно када се користи `evtest` апликација која је укључена и у root filesystem.

Само покрените:

```
evtest
```

Апликација ће вам приказати све доступне улазне уређаје и допустити вам да одаберете један по свом избору (обавезно одаберите један, иако је подразумевано 0 немојте само притиснути [Ентер], већ претходно упишите свој избор).

Можете одмах да тестирате и на следећи начин

```
evtest /dev/input/event0
```

Притисните различите тастере (комбинације) и уочите како су одговарајући догађаји пријављени кроз евтест апликацију.

САЧУВАЈТЕ СВЕ ИЗМЕНЕ

Да бисте потврдили и сачували све измене, најбоље је да их додате, а потом и локално комитујете на GIT, док сте позиционирани у неки од директоријума репозиторијума који је мењан, нпр. `~/linux-kernel-labs` и исто за `~/linux-kernel-labs/src/linux`:

```
git add -A

git commit -as -m "dan11.2 zavrsen"
```

Да би измене постале видљиве и у репозиторијуму на серверу, потребно би још било урадити нпр. `git push`, али то у овом случају није неопходно нити имамо неопходна права за то.