SCIENCES SORBONNE UNIVERSITÉ

About event-based programming

Fabrice.Kordon@lip6.fr

LIP6

Event?

Event

- Basis of event-based programming…
  … and thus basis for «reactive» applications

Interruptions

- «  » provoking a rerouting to an address
  - **Interruption, traps**
  - **Handled by hardware**

Exceptions

- «  » provoking a rerouting to a «handler»
  - **Higher level of abstraction (management of the stack)**
  - **Handled by the Operating System**
  - **Possible to stack exceptions**
- Unix signals = variants of exceptions

Event?

Event

- Basis of ev
  … anc

Interrupt

- «  » provoking a rerouting to an address
  - Interruption, traps
  - Handled by hardware

Exceptio

- «  » pro
  - Higher leve
  - Handled b
  - Possible to
- Unix signals = variants of exceptions

Differences

How the «handler» is specified

rerouting mechanisms

Graphical interfaces

typical example since the 1970's
(X by Xerox)

Call-back

Supported by the Operating System

- Enable an event (enable_event)
  - **Associate a handler procedure to an event**
- Deactivate an event (disable_event)
  - **Disable an handler/event association**
- Wait for an event (wait_event)
  - **Hand over to the OS**
- Efficient use of the CPU
  - **Better than active wait**

For unassociated events

- The OS does nothing
- Corresponding events are lost

As a conclusion

Smartphones & tablets

- User interface centered
- Reactive programming (as for user interfaces)
- Event programming (with dedicated frameworks)

Events are

- «Concurrenc
- Deep unders urces)
  - **e.g. event-loop**
- Event management based on high-level languages (objets oriented)
  - **e.g. Objective-C & Swift (iOS) or Java(Android)**

Next 2 videos…

Study of two typical examples