

**Lab #2:**  
**KP development with C\_KPI**

Instructor: Ivan Galov

# Smart-M3 KP development tools

## 1. Low-level programming tools

- Based on triples
- Basic manipulations

## 2. High-level programming tools

- Based on ontology entities
- Advanced manipulations

# Low-level programming tools

## 1. C\_KPI

Developer: **PetrSU** (*based on KPI\_Low from VTT*)

Language (Tool / KP): **ANSI C / C**

Interfaces: **TCP, Nota**

## 2. Whiteboard / Whiteboard-Qt + QML

Developer: **Nokia Corporation**

Language (Tool / KP): **C + Glib + Dbus / C**

Interfaces: **TCP, Nota**

## 3. Smart-M3 Java KPI library

Developer: **University of Bologna, VTT-Oulu**

Language (Tool / KP): **Java / Java**

Interfaces: **TCP**

## 4. M3 Python KPI (m3\_kp)

Developer: **Nokia Corporation**

Language (Tool / KP): **Python / Python**

Interfaces: **TCP**

## 5. C# KPI

Developer: **University of Bologna**

Language (Tool / KP): **C# / C#**

Interfaces: **TCP**

# High-level programming tools

## 1. **SmartSlog**

Developer: **PetrSU**

Language (Tool / KP): **Java, ANSI C / C, C#**

Interfaces: **TCP, Nota**

## 2. **Smart-M3 ontology to C-API generator**

Developer: **Abo DIEM**

Language (Tool / KP): **Java, C + Glib + Dbus / C**

Interfaces: **TCP, Nota**

## 3. **Smart-M3 ontology to Python generator**

Developer: **Nokia Corporation**

Language (Tool, KP): **Java, Python / Python**

Interfaces: **TCP**

# C\_KPI: installation

## 1. Check global variables export in ~/.bashrc

```
~> export PKG_CONFIG_PATH=$PKG_CONFIG_PATH:<inst_path>/lib64/pkgconfig
~> export LD_LIBRARY_PATH=$LD_LIBRARY_PATH:<inst_path>/lib64
```

## 2. Install scew

- Download [scew](#) (scew-1.1.7.tar.gz)

- Unpack

```
~> tar -zxf scew-1.1.7.tar.gz
```

```
~> cd scew-1.1.7
```

- Install

```
~> ./configure --prefix=$HOME/inst
```

```
~> make
```

```
~> make install
```

# C\_KPI: installation

## 3. Install C\_KPI

- Download [C\\_KPI](#) (C\_KPI.tar.gz)
- Unpack

```
~> tar -zxvf C_KPI.tar.gz
~> cd C_KPI
```
- Install

```
~> ./configure --prefix=$HOME/inst
~> make
~> make install
```

# C\_KPI: example

## Run example

- Run SIB
- Open HelloWorld demo folder  
~> `cd C_KPI/demos/HelloWorld`
- Edit SIB IP and port in *publisher\_kp.c* and *consumer\_kp.c* (line ~50):  
`#define KP_SS_ADDRESS "127.0.0.1"`  
`#define KP_SS_PORT 10010`
- Make and run (in different terminals)  
~> `make`  
~> `./publisher`  
~> `./consumer`

# C\_KPI

## 1. API

See file <ckpi.h>

## 2. Basic scenarios

- Initialize connection to Smart Space
- Send set of triples to Smart Space
- Receive set of triples from Smart Space
- Subscription to triples
- Asynchronous subscription



# C\_KPI: Initialize connection

```
ss_info_t ss_info;

/* Initializing connection parameters */
ss_init_space_info(&ss_info, "X",
                  "127.0.0.1", 10010);

if (ss_join(&ss_info, "KP name") == -1)
{
    /* Error handle */
}

/*
... KP logic ...
*/

ss_leave(&ss_info);
```

# C\_KPI: Send and remove triples

```
ss_triple_t * triple = NULL;
```

```
/* insert triples to the smart space */
```

```
ss_add_triple(&triple, "subject", "predicate", "object – another subject",  
             SS_RDF_TYPE_URI, SS_RDF_TYPE_URI);
```

```
ss_add_triple(&triple, "subject", "predicate", "object - value",  
             SS_RDF_TYPE_URI, SS_RDF_TYPE_LIT);
```

```
ss_insert(ss_info, triple, NULL);
```

```
ss_delete_triples(triple); /* clear memory */
```

```
triple = NULL;
```

```
/* remove triples from the smart space */
```

```
ss_add_triple(&triple, "subject", "predicate", SS_RDF_SIB_ANY,  
             SS_RDF_TYPE_URI, SS_RDF_TYPE_URI);
```

```
ss_remove(ss_info, triple, NULL);
```

```
ss_delete_triples(triple);
```

# C\_KPI: Receive (query) set of triples

```
ss_triple_t * sensor_info_rqst = NULL;
ss_triple_t * result_triple = NULL;

/* query for triples */
ss_add_triple(&sensor_info_rqst, "subject", "predicate",
              SS_RDF_SIB_ANY,
              SS_RDF_TYPE_URI, SS_RDF_TYPE_URI);
if(ss_query(ss_info, sensor_info_rqst, &result_triple) < 0)
{ /* Unable to query */ }

ss_delete_triples(sensor_info_rqst);

/* ... Handle result_triple ... */

ss_delete_triples(result_triple);
```

# C\_KPI: Subscription to triples

```
ss_triple_t * triple_rqst = NULL;          ss_triple_t * n_val = NULL;
ss_triple_t * triple = NULL;              ss_triple_t * o_val = NULL;

/* query for triples */
ss_add_triple(&triple_rqst, "subject", "predicate",
              SS_RDF_SIB_ANY, SS_RDF_TYPE_URI, SS_RDF_TYPE_URI);

if(ss_subscribe(ss_info, subs_info, triple_rqst, &triple) < 0)
{ /* Failed to subscribe */ }
ss_delete_triples(triple_rqst);

int status = ss_subscribe_indication(&ss_info, subs_info, &n_val,
&o_val, 3000);

if(status == 0) continue; /* timeout */
if(status < 0) { /* Error occurred */ }
if(status == 1) { /* ... new values ... */ }
```

# C\_KPI: Asynchronous subscription

```
pthread_t thread;
thread_param_t * param;

/* ... prepare to subscription and subscribe */

ss_add_triple(&triple_rqst, "subject", "predicate",
              SS_RDF_SIB_ANY, SS_RDF_TYPE_URI, SS_RDF_TYPE_URI);

param = (thread_param_t *) malloc(sizeof(thread_param_t));

param->ss_info = *ss_info;
param->subs_info = subs_info;

if (pthread_create(&thread, NULL,
                  subscribe_handler, (void *)param))
{ /* handle error */ }
```

# Task

- **First KP** should publish 5 triples
- **Second KP** should receive 3 of these triples
- You can change an example: “**HelloWorld**” demo

## Optional:

- Extend Second KP to **subscription**
- Extend Second KP to **asynchronous** subscription