GUIT *meeting* 2017

Let's Connect LuaTEX To The World

Roberto Giacomelli
giaconet.mailbox@gmail.com

Mestre, 21 ottobre 2017

Reporting nelle aziende

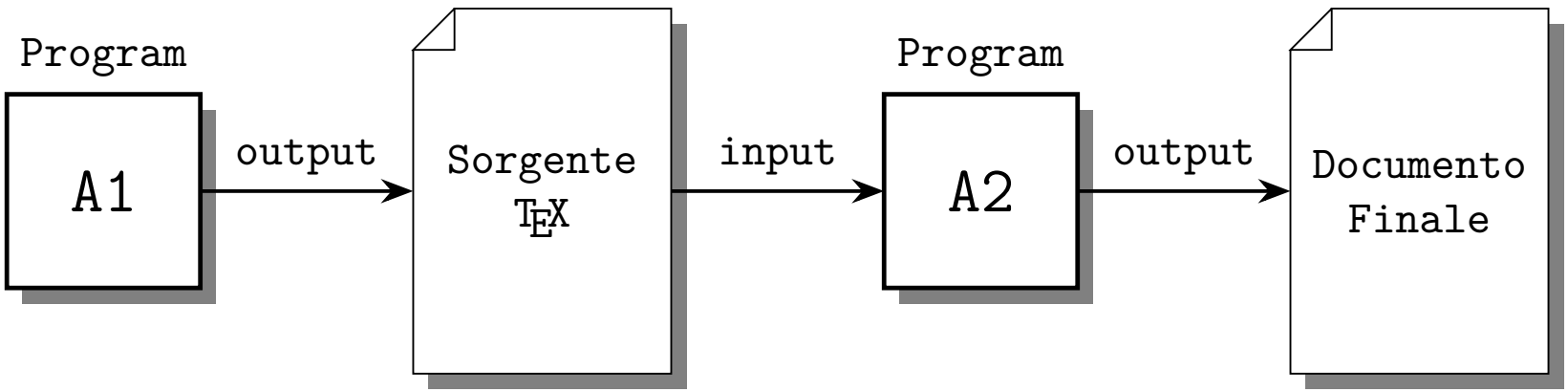
*Definizione di **report**: nell'attività aziendale o professionale, è la documentazione che richiede una qualche forma di elaborazione automatica.*

Per esempio:

- rendicontazioni economiche che richiedono calcoli finanziari;
- stato di progetti in corso con collezioni complesse di dati;
- inventari di magazzino i cui dati risiedono in database SQL;
- comunicazioni ai clienti, un compito ripetitivo;
- eccetera, eccetera, eccetera.

Costruzione automatica del sorgente

Il sistema TEX non è stato progettato per questi compiti, tuttavia è semplice costruire in modo automatico i sorgenti dei report, essendo loro semplici file di testo.



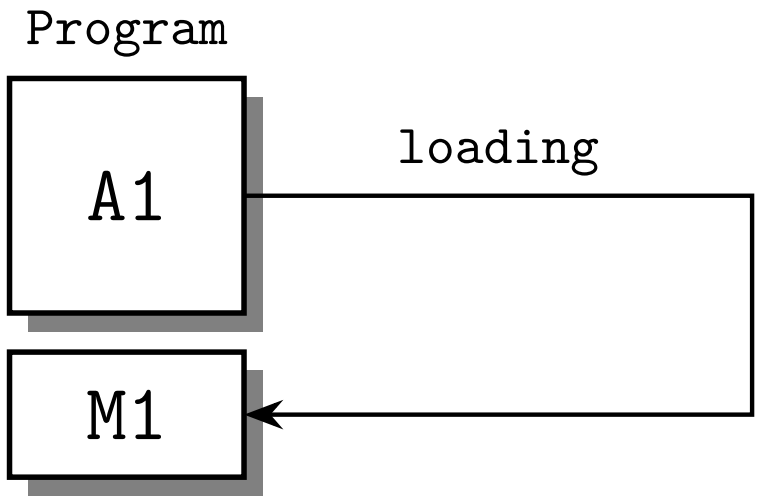
La tecnica ha il grande pregio di tenere separate le esecuzioni tra il componente `A1` e il compositore `A2` scelto della famiglia TEX.

Si veda l'articolo "*Generare documenti LATEX con diversi linguaggi di programmazione*" su ArsTEXnica n. 20 del 2015

Caricamento di componenti esterni

Con l'ingresso di LuaTEX si sono aperti ampi e nuovi scenari.
Lua è un linguaggio dinamico scritto in C, progettato per estendere applicazioni, e per essere estensibile:

- tecnologia FFI Foreign Function Interface;
- API di Lua;
- loading di librerie dinamiche in formato binario ABI.



Per FFI si veda anche l'articolo "*A Database Experiment With LuajitLATEX*" su ArsTEXnica n. 23 del 2017

Scambio di messaggi: ZeroMQ + LuaTEX

Caricando un'unica libreria è possibile costruire una *rete* dove LuaTEX *comunica* con gli altri nodi per mezzo di *messaggi*.

Il progetto `http://zeromq.org/` è uno dei candidati migliori. È una libreria in C++ che è possibile usare in LuaTEX.

Lo scambio di messaggi è bidirezionale mentre i nodi della rete sono componenti del tutto *separati*.

Scambio di messaggi: client/server

Durante la compilazione di un sorgente `.tex` LuaTEX può usufruire di servizi esterni inviando al loro indirizzo messaggi di richiesta e ricevendo in cambio messaggi con i dati elaborati.

Un server d'esempio: il doppio di un numero

Il server scritto in Rust[1] raddoppia un intero:

```
extern crate zmq;
fn main() {
    let ctx = zmq::Context::new();
    let responder = ctx.socket(zmq::REP).unwrap();
    assert!(responder.bind("tcp://192.168.0.252:5555").is_ok());
    let mut msg = zmq::Message::new().unwrap();
    println!("Server ready...");
    loop {
        responder.recv(&mut msg, 0);
        // message decoding
        let n: i32 = msg.as_str().unwrap().parse().unwrap();
        println!("Received [{}]", n);
        // reply
        responder.send_str(&(2 * n).to_string()[..], 0).unwrap();
    }
}
```

[1] www.rust-lang.org

Il client in LuaTEX

Il sorgente `.tex` per LuaTEX che utilizza il server:

```
% !TeX program = LuaTeX-shell-escape
\input luapackageloader.sty
\directlua{
    local zmq = require "lzmq"
    local context = zmq.init(1)

    local socket = context:socket(zmq.REQ)
    socket:connect("tcp://localhost:5555")

    local n = 28
    socket:send(tostring(n))
    local reply = socket:recv()
    tex.print("Received... [" .. reply .. "]")
    socket:close()
    context:term()
}
\bye
```

QRCode 1/2 — i messaggi multiparte di ZeroMQ

Poiché i messaggi sono pure sequenze di byte, occorre stabilirne un formato per codificare e decodificare l'informazione ad alto livello.

In questo esempio, il formato dei messaggi di richiesta è:

```
Frame 0: "QRCODE"
Frame 1: <text>
Frame 2: "END"
```

I messaggi del server di risposta sono invece:

```
Frame 0 : "QRCODE"
Frame 1 : <qrcode 0/1 sequence>
Frame 2 : "END"
```

oppure:

```
Frame 0 : "ERR"
Frame 1 : <error description>
Frame 2 : "END"
```

QRCode 2/2

Scrivendo un'opportuna libreria in Lua è possibile utilizzare il server che codifica il QRCode di un testo in modo semplice.

```
% !TeX program = LuaTeX--shell-escape
\directlua{
local libmsg = require "libmsg"
local QRCodeSrv, err = libmsg:service "tcp://localhost:5556"
local dataset, err = QRCodeSrv:send_and_recv{
"QRCODE", "http://www.guitex.org/home/it/forum/recent-topics", "END"
}
% ... ulteriori elaborazioni su dataset
}
% ... stampa del disegno del simbolo QRCode
\bye
```

(trovate il codice completo nell'articolo)

Numeri primi 1/3 — server multifunzione

Il messaggio di richiesta può contenere una *chiave* che individua una corrispondente elaborazione. L'invio del messaggio al server appare come una sorta di chiamata di funzione remota.

`PRIME-CHECK:` test di primalità;

`PRIME-COUNT:` conteggio dei primi in un intervallo;

`PRIME-LIST:` lista dei primi in un intervallo.

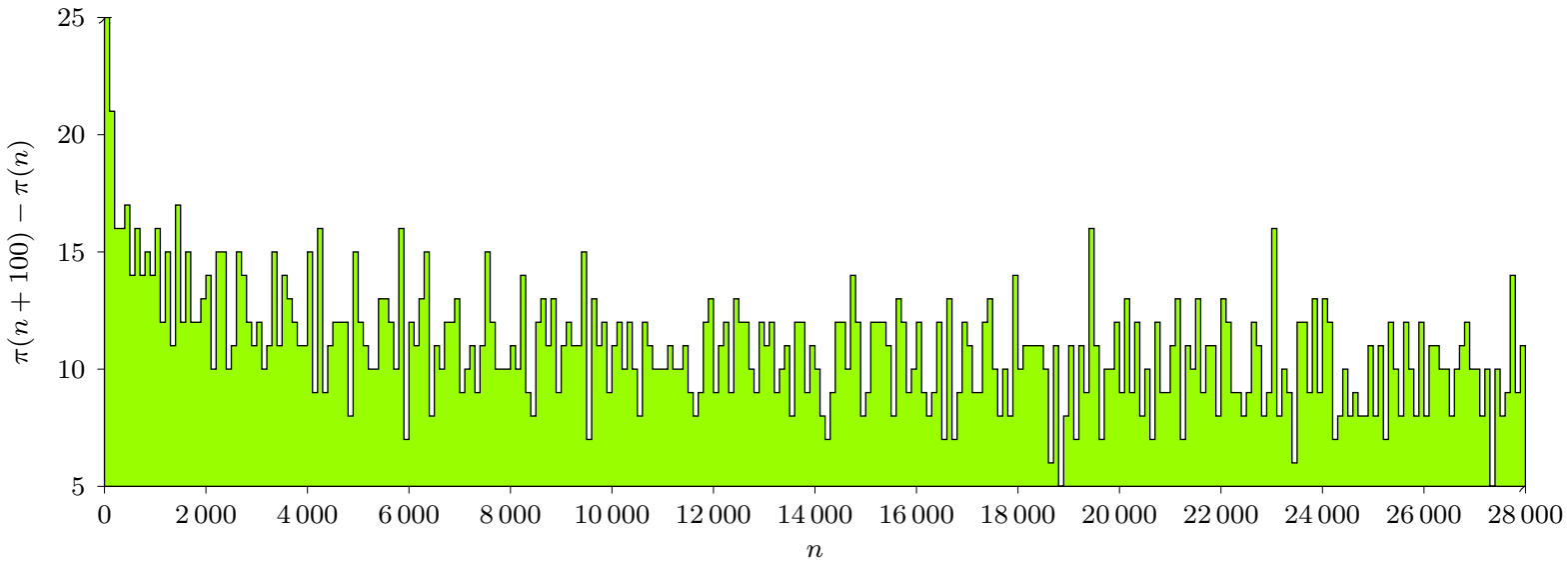
Per esempio, il messaggio di richiesta potrebbe essere nel formato:

```
Frame 0: <function ID>
Frame 1: <arg1>
Frame 2: <arg2>
...
Frame n: "END"
```

Numeri primi 2/3

Un sorgente LuaTEX inviando al server messaggi con la chiave `PRIME-COUNT`, può produrre un grafico con il pacchetto *pgfplots*:

```
% !TeX program = LuaLaTeX--shell-escape
... \directlua{
    local libmsg = require "libmsg"
    PrimeSrv = libmsg:service "tcp://localhost:5557"
    for _ = 1, points do
        local tprime, err = PrimeSrv:call("PRIME-COUNT", a, a+step-1)
        ...
    end
} ...
```



Numeri primi 3/3

o produrre la tabella che elenca i primi inferiori di 1 000:

```
% !TeX program = LuaLaTeX--shell-escape
...\directlua{
    local libmsg = require "libmsg"
    PrimeSrv = libmsg:service "tcp://localhost:5557"
    ...
    local list, err = PrimeSrv:call("PRIME-LIST", 1, 1000)
    ...
} ...
```

2	3	5	7	11	13	17	19	23	29	31	37	41	43
47	53	59	61	67	71	73	79	83	89	97	101	103	107
109	113	127	131	137	139	149	151	157	163	167	173	179	181
191	193	197	199	211	223	227	229	233	239	241	251	257	263
269	271	277	281	283	293	307	311	313	317	331	337	347	349
353	359	367	373	379	383	389	397	401	409	419	421	431	433
439	443	449	457	461	463	467	479	487	491	499	503	509	521
523	541	547	557	563	569	571	577	587	593	599	601	607	613
617	619	631	641	643	647	653	659	661	673	677	683	691	701
709	719	727	733	739	743	751	757	761	769	773	787	797	809
811	821	823	827	829	839	853	857	859	863	877	881	883	887
907	911	919	929	937	941	947	953	967	971	977	983	991	997

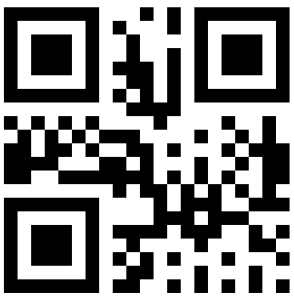
Database 1/2 — accesso a più servizi

Stampa di badge degli impiegati, dal sorgente LuaTEX doppio accesso a un database SQLite e alla codifica QRCode:

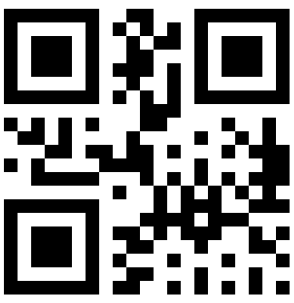
```
% !TeX program = LuaLaTeX--shell-escape
...\directlua{  local libmsg = require "libmsg"
    DBSrv, err = libmsg:service "tcp://localhost:5558"
    local employees, err = DBSrv:call("QUERY",
    [[SELECT employees.emp_no AS empid,
    employees.first_name || ' ' || employees.last_name AS name,
    employees.birth_date    AS birthdate,
    employees.hire_date     AS hiredate,
    departments.dept_name   AS dept,
    dept_emp.from_date      AS fromdate
    FROM departments, dept_emp, employees
    WHERE departments.dept_no = dept_emp.dept_no AND
    employees.emp_no = dept_emp.emp_no ORDER BY empid LIMIT ?;]], 2)
    ...
    QRCodeSrv, err = libmsg:service "tcp://localhost:5556"
    local tmsg, err = QRCodeSrv:call("QRCODE", empid) ...
}...
```

Database 2/2

Risultato: *i badge degli impiegati*



ID: 10001
Name: Georgi Facello
Birth Date: 1953-09-02
Dept.: Development
Hire date: 1986-06-26
From: 1986-06-26



ID: 10002
Name: Bezalel Simmel
Birth Date: 1964-06-02
Dept.: Sales
Hire date: 1985-11-21
From: 1996-08-03

Conclusioni

Vantaggi del processo:

- incremento della qualità tipografica e delle capacità di evoluzione dei report;
- integrazione con i sistemi informativi aziendali nello scenario a nodi distribuiti.

Conseguenze della tecnologia ZeroMQ:

- il caricamento di librerie binarie in LuaTEX/LuajitTEX si limita al solo accesso a ZeroMQ;
- per lo sviluppo dei server è possibile usare qualsiasi linguaggio che acceda a ZeroMQ (e sono davvero tanti);
- lo sviluppo del codice entra definitivamente nel campo dell'ingegneria del software.

Risorse e ringraziamenti

Principali risorse sulle tecnologie trattate:

- **LuaTEX manual**: `> texdoc luatex`
- **ZeroMQ Guide**: `http://zguide.zeromq.org/page:all`
- **The Rust Book**: `https://doc.rust-lang.org/book/`

Grazie per l'attenzione

e grazie mille Mestre!

Domande?