PB173 – Binární programování Linux

XII. Komunikace mezi procesy, část 2

Jiri Slaby

Fakulta informatiky
Masarykova univerzita

10. 12. 2013

Komunikace mezi procesy

Alespoň 2 procesy, které chtějí komunikovat

Minule

- Roura (`pipe`)
- Sdílená paměť (`mmap`)
- Plná meziprocesová komunikace (IPC)

Dnes

1. Netlink
2. RPC
   - RPC rozhraní pro server
   - RPC rozhraní pro klienta
   - XDR

Sekce 1

Netlink

Netlink

- Protokol pro komunikaci s jádrem
  - Standardní socket (`sys/socket.h`)
  - Ale umožňuje i komunikaci mezi procesy
- Adresování: `struct sockaddr_nl` (`linux/netlink.h`)
  - `nl_family = AF_NETLINK`
  - `nl_pid` je většinou číslo procesu
- Vytvoření: `socket(AF_NETLINK, SOCK_RAW, NETLINK_USERSOCK)`
- Server
  - Provede `bind` s nastaveným `sockaddr_nl.nl_pid`
  - Potom poslouchá pomocí `recvmsg`
- Klient
  - Posílá pomocí `sendmsg`
- Zpráva: `struct nlmsghdr`
  - Operace: `NLMSG_DATA` (data), `NLMSG_PAYLOAD` (délka), ...

Úkol

Vytvoření netlink komunikace (bez hlaviček, jen `char *`)

1. Otevřete si kostru v `pb173-bin/12/netlink.c`
2. Doplňte server a klient (tři TODO)
3. V serveru proveďte:
   - `bind` s nastaveným `sockaddr_nl.nl_pid`
   - A v cyklu opakujte `recv` a `write` na standardní výstup
4. V klientovi proveďte:
   - `sendto` nějakých dat
5. Vyzkoušejte
   - Vypsaný PID použijte jako adresu v klientovi

Sekce 2

RPC

RPC

- Vzdálené volání procedur
  - Volání funkcí přes síť
  - UDP nebo TCP
  - Předávání parametrů a návratových hodnot
  - Využití např. v NFS
- Portmapper
  - Lokální služba
  - Mapuje číslo „programu” (funkce), jeho verzi a protokol na funkce
    - Registrovaná čísla v `/etc/rpc`

RPC rozhraní pro server

- `rpc/rpc.h`, `man 3 rpc`
- Vytvoření spojení: `svctcp_create`, `svcudp_create`
  - Parametr: socket nebo `RPC_ANYSOCK`
- Registrace volatelného „programu” (funkce): `svc_register`
  - Registruje program u portmapperu pomocí `pmap_set`
  - Viditelné v `rpcinfo -p`
  - Je zvykem před tím volat `pmap_unset`
    - Zruší případnou zastaralou registraci
  - Volá se funkce udaná parametrem `dispatch`
    - Parametr `svc_req->rq_proc` určuje funkcionalitu
- Obslužná smyčka: `svc_run`

RPC rozhraní pro server

Příklad

```
#define RUSERSPROG 0x40000000
#define RUSERSVERS 1

static void dispatch(struct svc_req *req, SVCXPRT *xprt)
{
   printf ("request %ld\n", req->rq_proc);
}

int main(void) {
  SVCXPRT *xprt = svctcp_create(RPC_ANYSOCK, 0, 0);
  svc_register(xprt, RUSERSPROG, RUSERSVERS, dispatch, IPPROTO_TCP);
  svc_run();

  return 0;
}
```

Úkol

Vytvoření RPC serveru

1. Vytvořte si TCP spojení (`svctcp_create` a `RPC_ANYSOCK`)
2. Zvolte si čísla programu a verze
   - Vyhněte se číslům programů z `/etc/rpc`
3. Zrušte stará mapování (`pmap_unset`)
4. Registrujte službu (`svc_register`)
   - V `dispatch` funkci proveďte výpis `svc_req->rq_proc`
5. Spusťte smyčku (`svc_run`)
6. Spusťte program
   - Za předpokladu, že běží portmapper

RPC rozhraní pro klienta

- Vytvoření spojení: `clnt_create`
  - Stroj, protokol a číslo programu+verze
- Zrušení spojení: `clnt_destroy`
- Zavolání programu (funkce): `clnt_call`
  - 2. parametr do `dispatch` jako `svc_req->rq_proc`
  - 3. a 5. parametr: kódovací funkce (zatím `xdr_void`)
  - 4. a 6. parametr: parametry vzdálené funkce (zatím `NULL`)
  - Timeout: čas na odpověď

Příklad

```
struct timeval tout = { 1, 0 };
CLIENT *cln = clnt_create("localhost", RUSERSPROG, RUSERSVERS, "tcp");
clnt_call(cln, 1, (xdrproc_t)xdr_void, NULL, (xdrproc_t)xdr_void, NULL, tout);
clnt_destroy(cln);
```

Úkol

Vytvoření RPC klienta

1. Vytvořte si TCP spojení na localhost (`clnt_create`)
   - Použijte čísla zvolená v serveru
2. Zavolejte několikrát `clnt_call`
   - S různým 2. parametrem
3. Spusťte program
   - Za předpokladu, že server běží

RPC a parametry

XDR

- `rpc/xdr.h`, `man 3 xdr`
- Funkce k přenositelnému zakódování dat
  - `xdr_void`
  - `xdr_char`, `xdr_short`, `xdr_int`, `xdr_long`
  - `xdr_float`, `xdr_double`
  - `xdr_string`, `xdr_array`
  - `...`
- Klient
  - Funkce shora jako 3. a 5. parametr pro `clnt_call`
  - Ukazatele na proměnné potom jako 4. a 6.
- Server
  - V `dispatch`: `svc_getargs`
  - A lze také odpovědět na zprávu: `svc_sendreply`

Úkol

Příklad

```
/* server */
int in;
svc_getargs(xprt, (xdrproc_t)xdr_int, (char *)&in);
in++;
svc_sendreply(xprt, (xdrproc_t)xdr_int, (char *)&in);

/* client */
int in = 5;
clnt_call (cln, 0, (xdrproc_t)xdr_int, (char *)&in, (xdrproc_t)xdr_int, (char *)&in, tout);
/* 'in' is 6 now (unless timed out) */
```

Předávání parametrů

1. V serveru vytvořte program
   - Vezme `int`, vynásobí `10` a vrátí `long`
2. Z klienta zavolejte
3. Vyzkoušejte