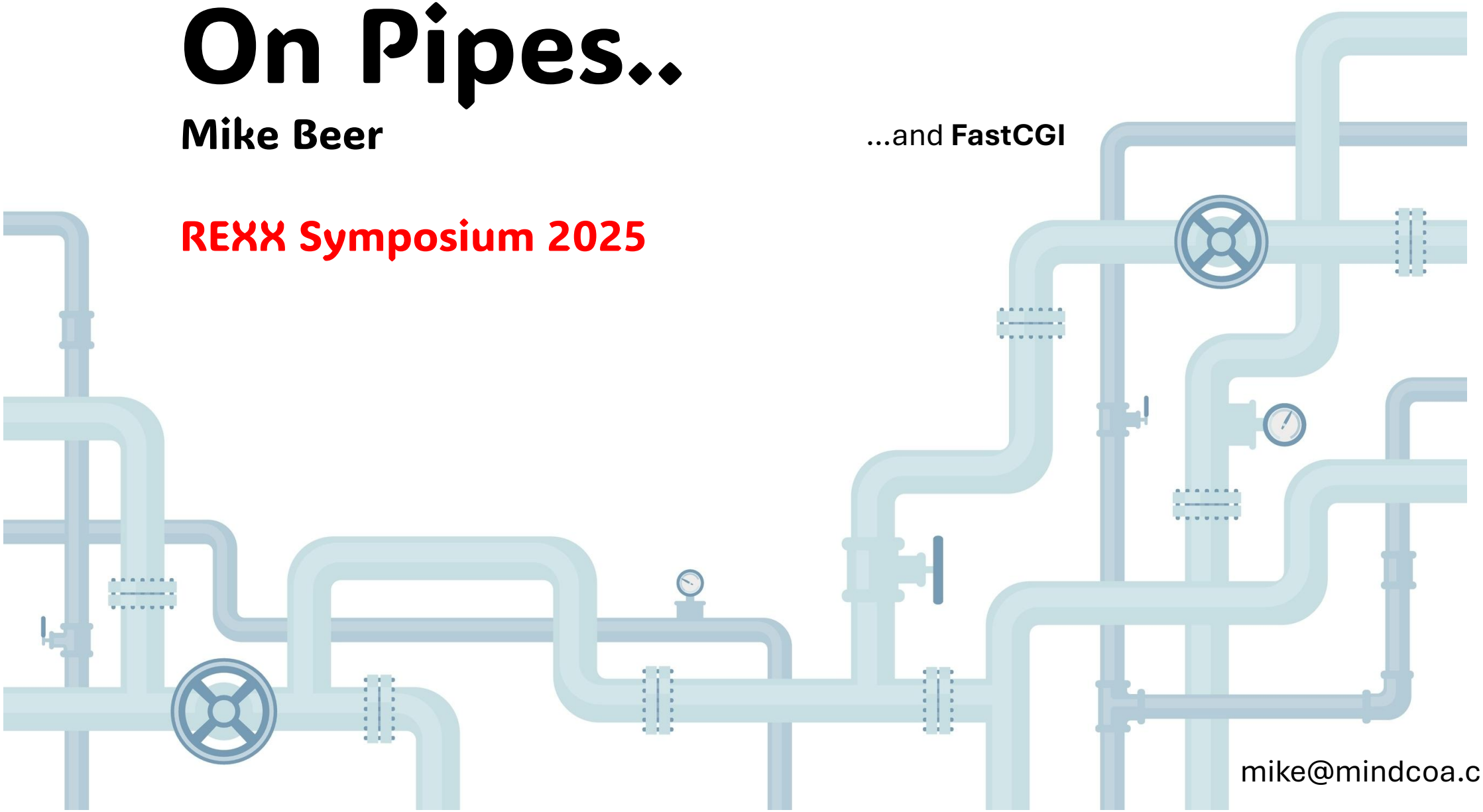
On Pipes..

Mike Beer

...and **FastCGI**

REXX Symposium 2025



mike@mindcoa.ch

AGENDA

- In the Beginning
- The Turn of a friendly Card
- The Race
- Where do we go from here?

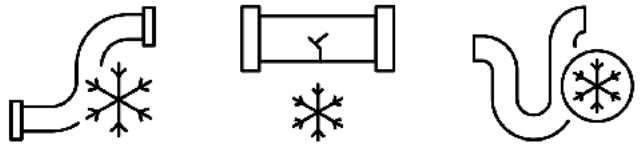
UNIX

- 3 Main features
  - PIPES (1972 McIlroy => Ken Tompson)
  - Hierarchical file system
  - Regular Expressions

https://softpanorama.org/Scripting/pipes.shtml

WHAT IS A PIPE?

dir *.rex | find „2025“



command 1 | command 2 | ...

STAGE

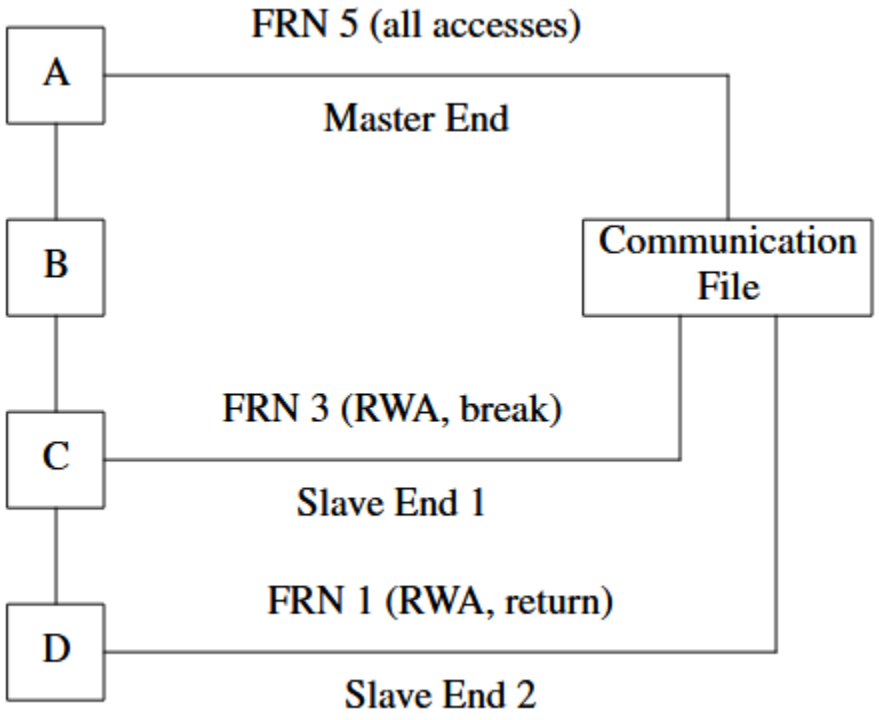
M. Douglas McIlroy: Communication Files: Interprocess IO before Pipes

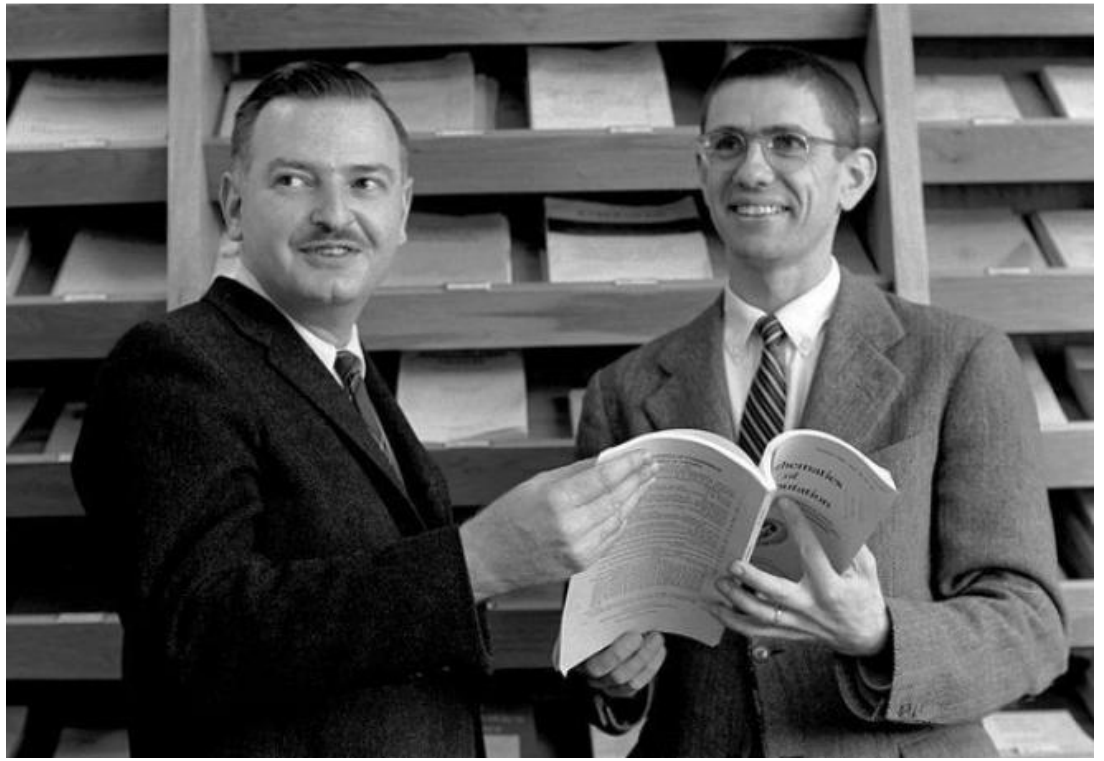
"Some time after the introduction of pipes to Unix, we in the Bell Labs Unix lab learned that the Dartmouth Time-Sharing System (**DTSS**) had a **mechanism for process-to-process IO called communication files.** [...]

Communication files were much more complicated than Unix pipes. They were also more powerful. Pipes could be simulated by communication files, but not vice versa. A pipe can handle neither the two-way communication nor the out-of-band signaling that communication files support."

https://www.cs.dartmouth.edu/doug/DTSS/commfiles.pdf

DTSS
Dartmouth Time-Sharing SYSTEM





https://www.cs.dartmouth.edu/~doug/DTSS/DTSSchapter5.pdf

Kemeny/Kurtz – BASIC
Ken Lochner (Com. Files) – 1960s

IBM PIPE HISTORY

- John Hartmann start of development of *CMS Pipelines* in 1980.
- Separate product during the 80's
- Integrated in VM/ESA late 1991.
- Freeze the 1.1.10 level in VM/ESA 2.3 in 1997.
- Included again since z/VM 6.4 (November 11, 2016).
- *BatchPipes/MVS (CMS Pipelines* for TSO) released in 1995

https://www.vm.ibm.com/pipelines/overview.html

http://vm.marist.edu/~pipeline/index.html

https://www.ibm.com/docs/en/SSB27U_7.4.0/pdf/c2462521.pdf

CMS

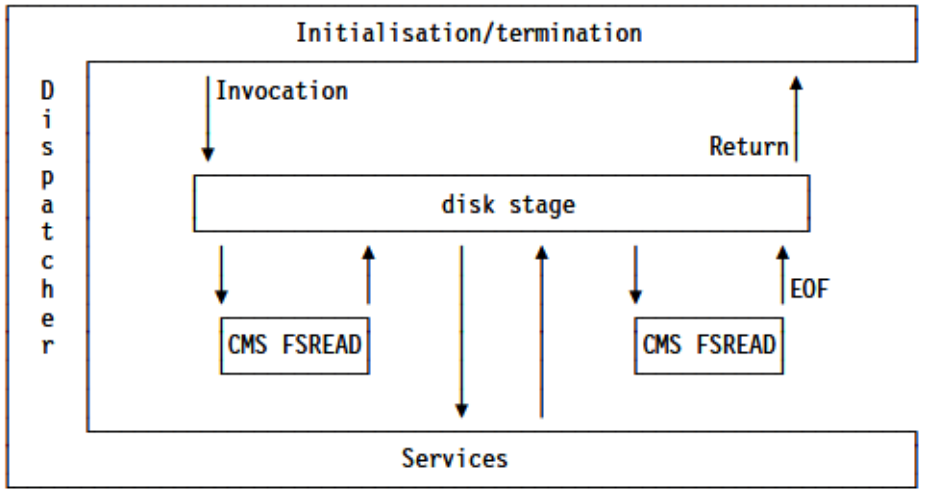
CMS Pipelines **Explained**

John P. Hartmann, IBM Danmark A/S

SHARE 88

VM Cluster Session 9112

Figure 2. One Stage's Interactions with the Pipeline Dispatcher



https://www.grothoff.org/christian/teaching/2008/4704/pipjarg.pdf

Plunging into Pipes

UIC Computer Center

Document #4617004

33 Pages

PLUNGING INTO PIPES

Melinda Varian

Office of Computing and Information Technology
Princeton University
87 Prospect Avenue
Princeton, NJ 08544 USA
--.--
BITNET: MAINT@PUCC
Internet: maint@pucc.princeton.edu
Telephone: (609) 258-6016

SHARE Summer 1992 Meeting
Session O700
August, 1992

„CMS Pipelines
is the most significant enhancement
to CMS since REXX.“

https://softpanorama.org/Scripting/Piporama/Reprints/v4617004.txt

Why another version?

- Rewriting File-Conversions (e.g. CSV)
- Improved EXECIO
- Compare REGINA, OOREXX, CREXX
- REXXTRY for AI and IoT?

Console I/O

pipe.rex "cons | sort | unique | cons"

INPUT DATA:

a

b

a

c

c

b

a

a

b

c

pipe.rex "LIT a | lit b | cons"

a

b

File I/O

Type abc.txt

aaa

b bb bb

cc c cccc

< OPEN READ REPLACE

<< OPEN APPEND

> WRITE REPLACE

>> WRITE APPEND

pipe.rex "<abc.txt | cons"

aaa

b bb bb

cc c cccc

pipe.rex "<abc.txt | reverse | >out.txt"

type out.txt

cc c cccc

b bb bb

aaa

Sample

pipe.rex "cons | xlate upper | locate /A/ | cons"

INPUT DATA:

a

abc

x

ddd

a

A

ABC

A

2 Types of Data

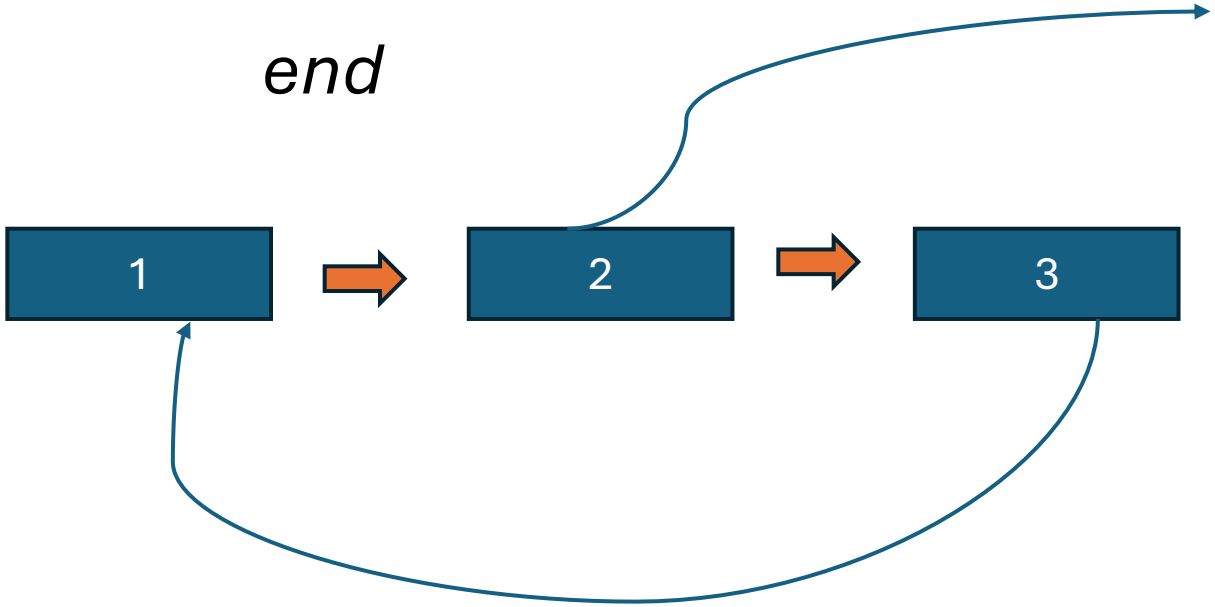
Data within stage(s)

Control structure of a pipe

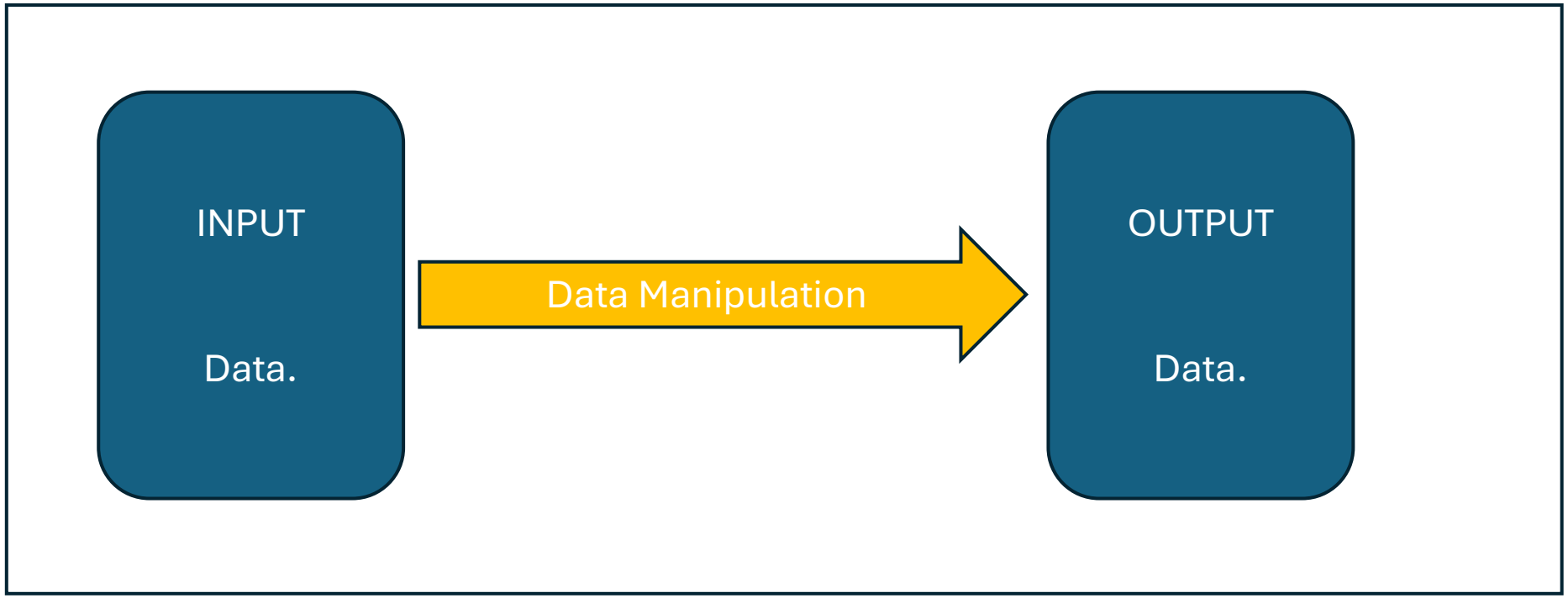
Variables and loops
IF .. GOTO .. IFZERO var GOTO ..

```
let a=3
loop: say $a.
dec a
ifzero a goto end
goto loop
end: say end
```

```
3
2
1
end
```

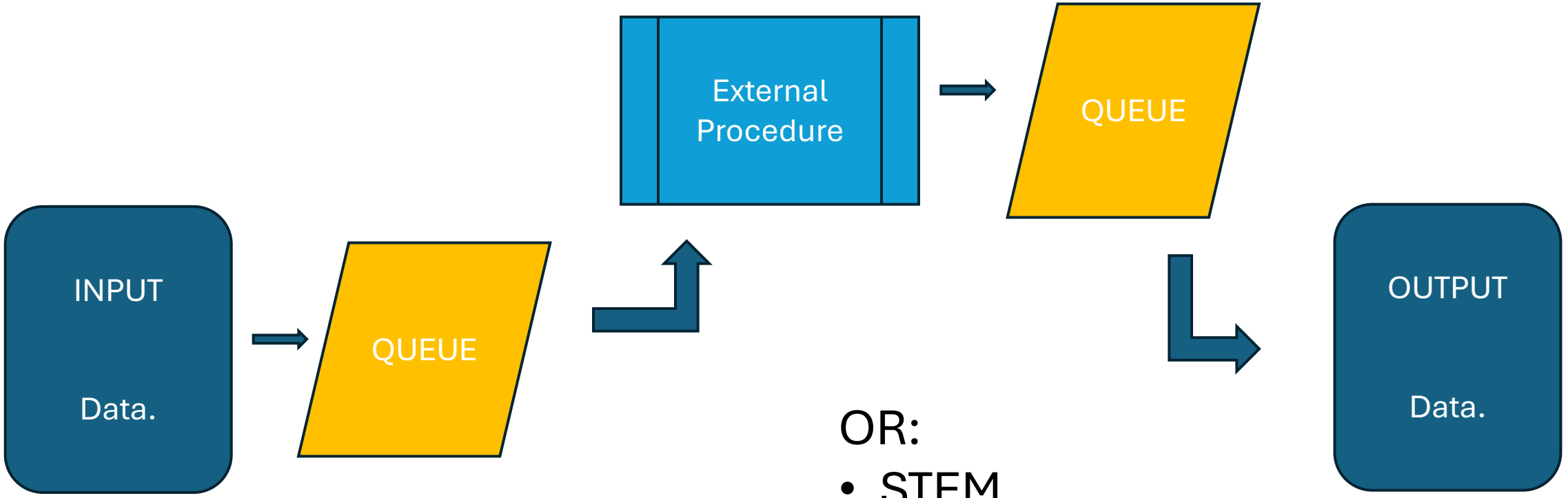


Data Flow internal: STEMS



STAGE

Data Flow external REXX: QUEUE



OR:

- STEM
- LOAD

Concepts
Words vs. Columns

SORT

wN - sort by word n

wN . - sort by word n using "." as delimiter instead of space

N sort by column N

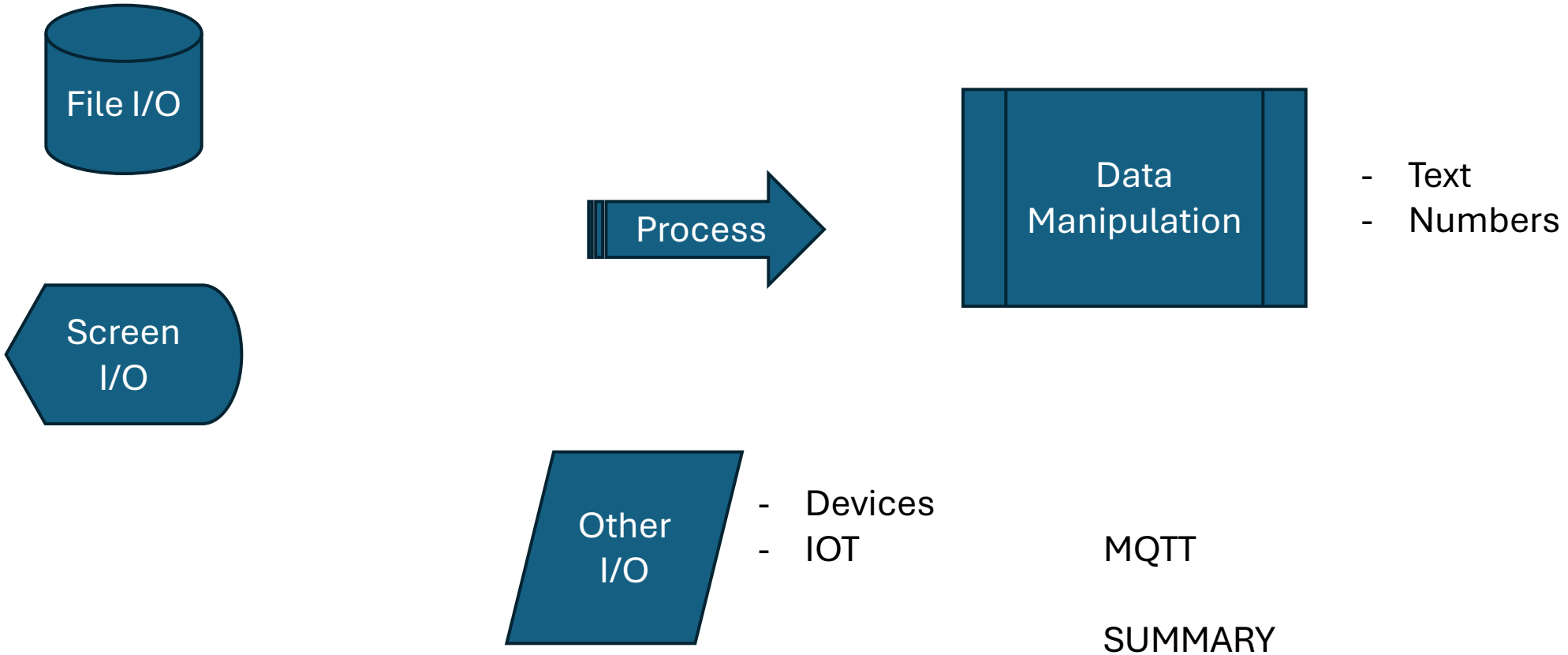
a-b sort using substring of columns a to b

IPC with calling REXX function

```
/*  ptest.rex */
s.1="a"
s.2="b"
s.3="c"
s.0=3
say pipe("LOAD s | cons")
say

queue 1
queue 2
say pipe("LOAD | cons")
```

```
ptest.rex
a
b
c
0

1
2
0

LOAD s | reverse | STEM s
```

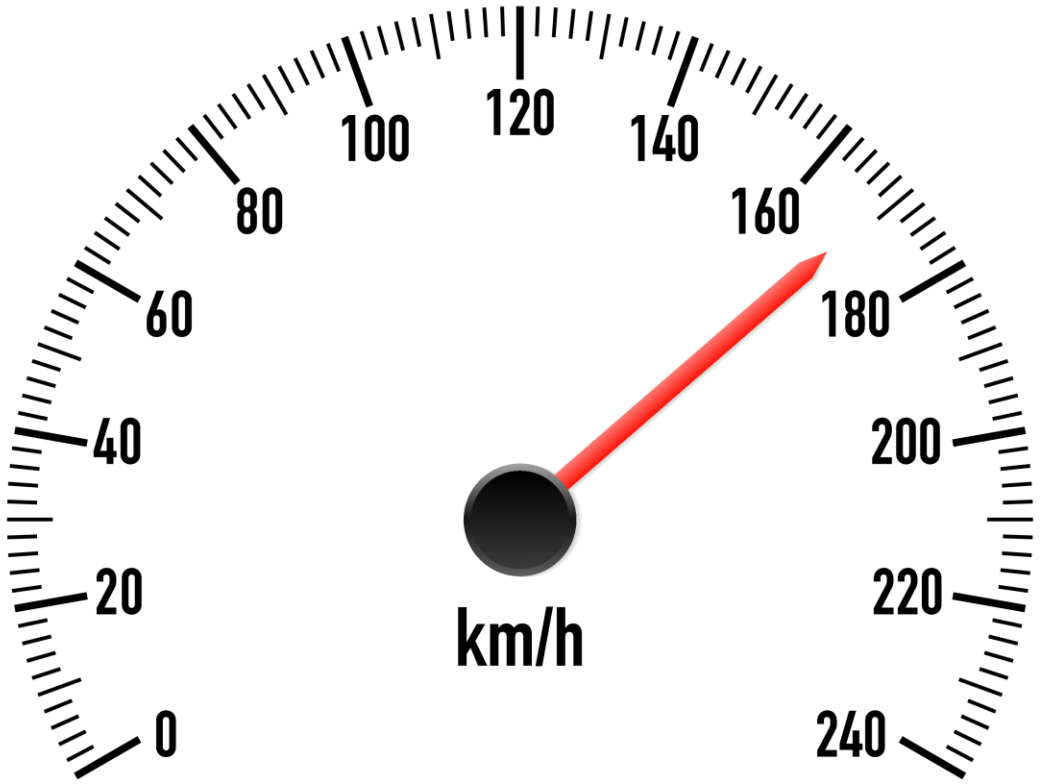
PIPE Components



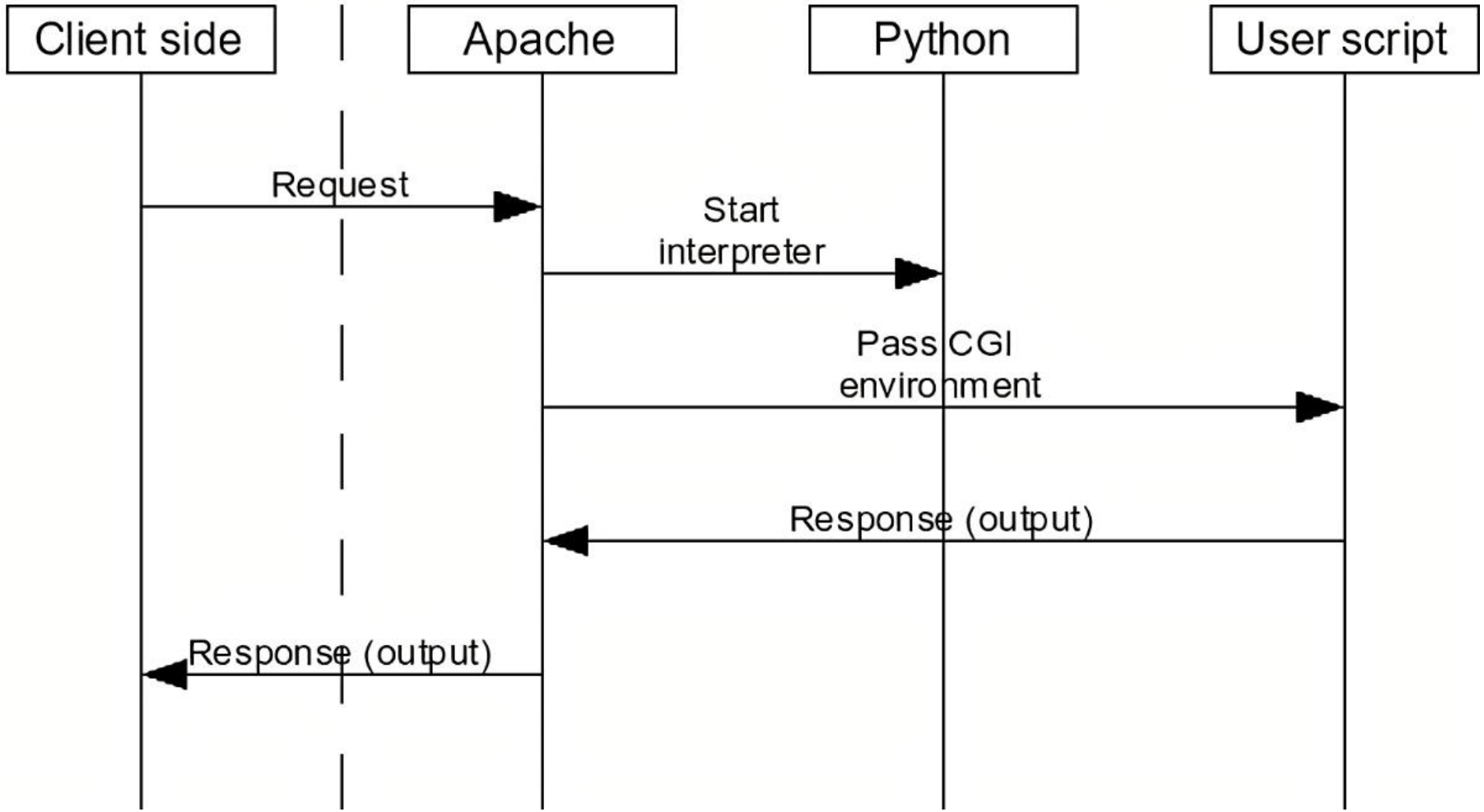
And many more functions...

STATE	HEAD
MAIL	TAIL
CURL	TAKE
BASE64	DROP
SPLIT	FRLABEL
JOIN	TOLABEL
CHANGE..	SPECS..

FastCGI



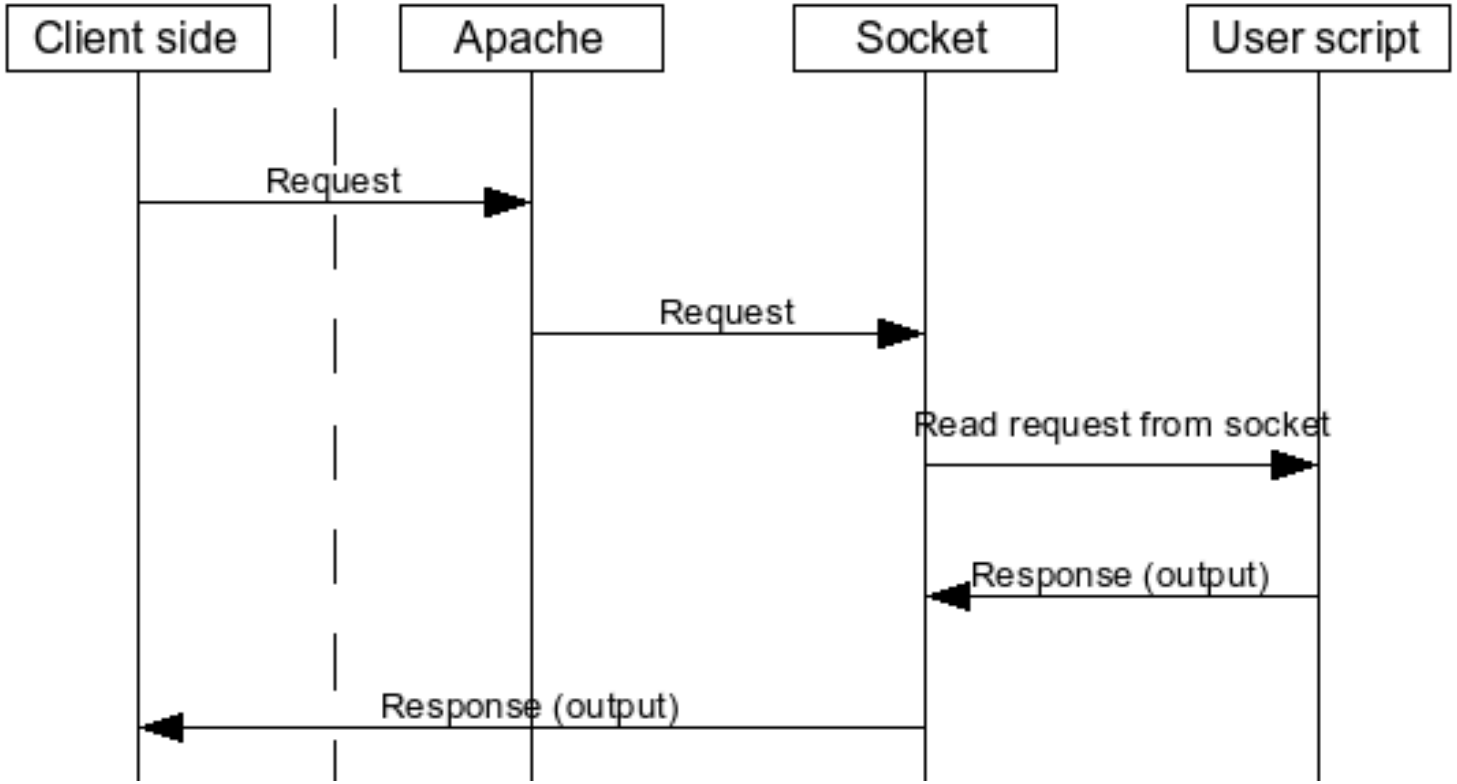
Web Server: CGI



Web Server: FastCGI Startup

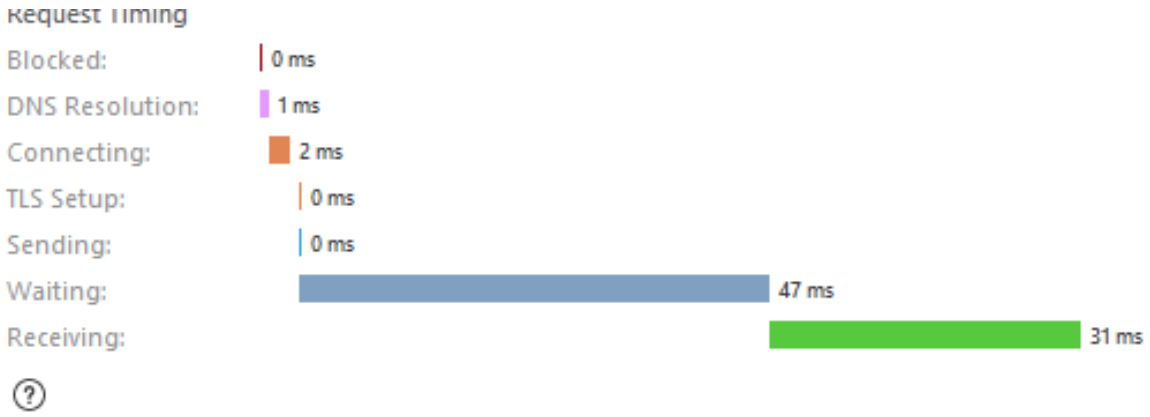
1) Start FastCGI REXX-Function
2) Create communication socket
3) Wait for Traffic from Apache

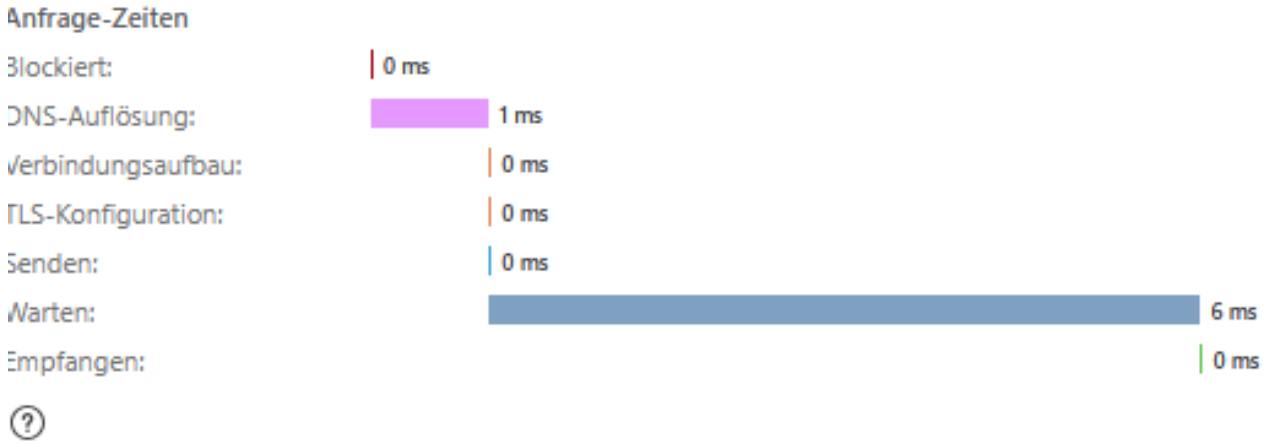
Web Server: FastCGI Operations



1) REXX app waits for connection
2) Performs action
3) Returns result via socket

Speed comparison





	CGI	FastCGI
Time	81 ms	7 ms

FastCGI handler

```
C:\Users\MikeBeer\apache\Apache24\cgi-bin>fastcgi.rexx
*********************************************
*                                           *
*      fastcgi/REXX up and running          *
*                                           *
*********************************************
socket 176 accepted

Version: 1
Type:    4
ID:      1
len:     1630
padlen:  0
TYPE=4
HTTP_HOST  :  localhost
HTTP_USER_AGENT  :  Mozilla/5.0 (Windows NT 10.0; Win64; x64; rv:138.0)
Gecko/20100101 Firefox/138.0
```

Apache httpd.conf
Requires mod_fcgi !!!

```
LoadModule proxy_module modules/mod_proxy.so

LoadModule proxy_fcgi_module modules/mod_proxy_fcgi.so


ProxyPassMatch "^/(.*\.rexx(/.*)?)$" "fcgi://127.0.0.1:8000/Users/MikeBeer/htdocs"
enablereuse=on
ProxyPassMatch "^/(.*\.rex(/.*)?)$" "fcgi://127.0.0.1:9000/Users/MikeBeer/htdocs"
enablereuse=on
```

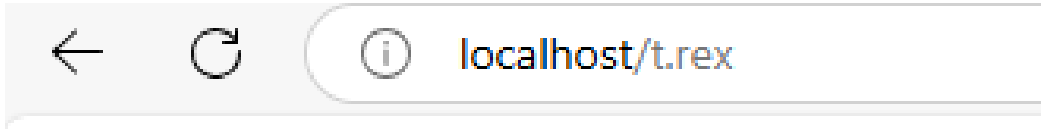
FastCGI handler – main loop

```
do forever
   ns = SockAccept(socket)
   if ns = -1 then do
      say "SockAccept() failed" sockpsock_errno()
      exit 6;
   end
   say "socket" ns "accepted"
   bytes  = SockRecv(ns, 'buf', BUFSIZE)
   if bytes  = -1 then do
      say "SockRecv() failed" sockpsock_errno()
      exit 7;
    end

    do while length(buf)>0
       ret = read_fastcgi(ns)     /* read one record */
       parse var ret type id
    end

end
```

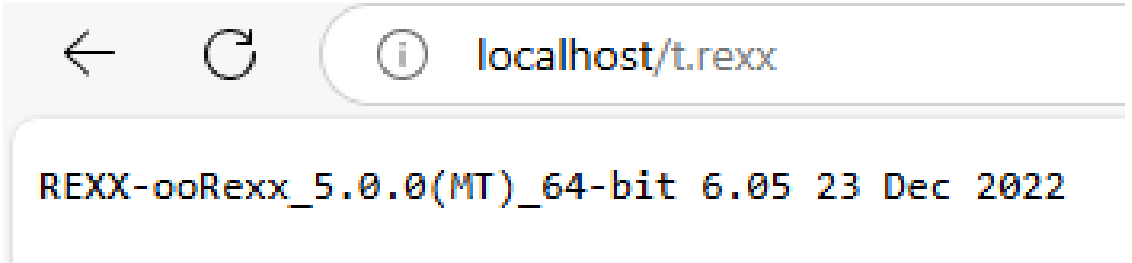
FastCGI handler – TYPE 1

```
select
   when type=1 then do                                   /* BEGIN REQUEST */
      s = left(buf,8)
      buf=substr(buf,9)
      role  = c2d(substr(s,1,1))*256 + c2d(substr(s,2,1))
      flags = c2d(substr(s,3,1))
      if wordpos(id,requests)=0 then requests=requests id
      sock.id = client_socket
      say "TYPE=1, ROLE="role "FLAGS="flags
      end

   when type=4 then do                                    /* FCGI_PARAMS */
```

```
00000 * * * Top of File * * *
00001 /* t.rex */
00002 parse version ret
00003 return ret
00004 * * * End of File * * *
```



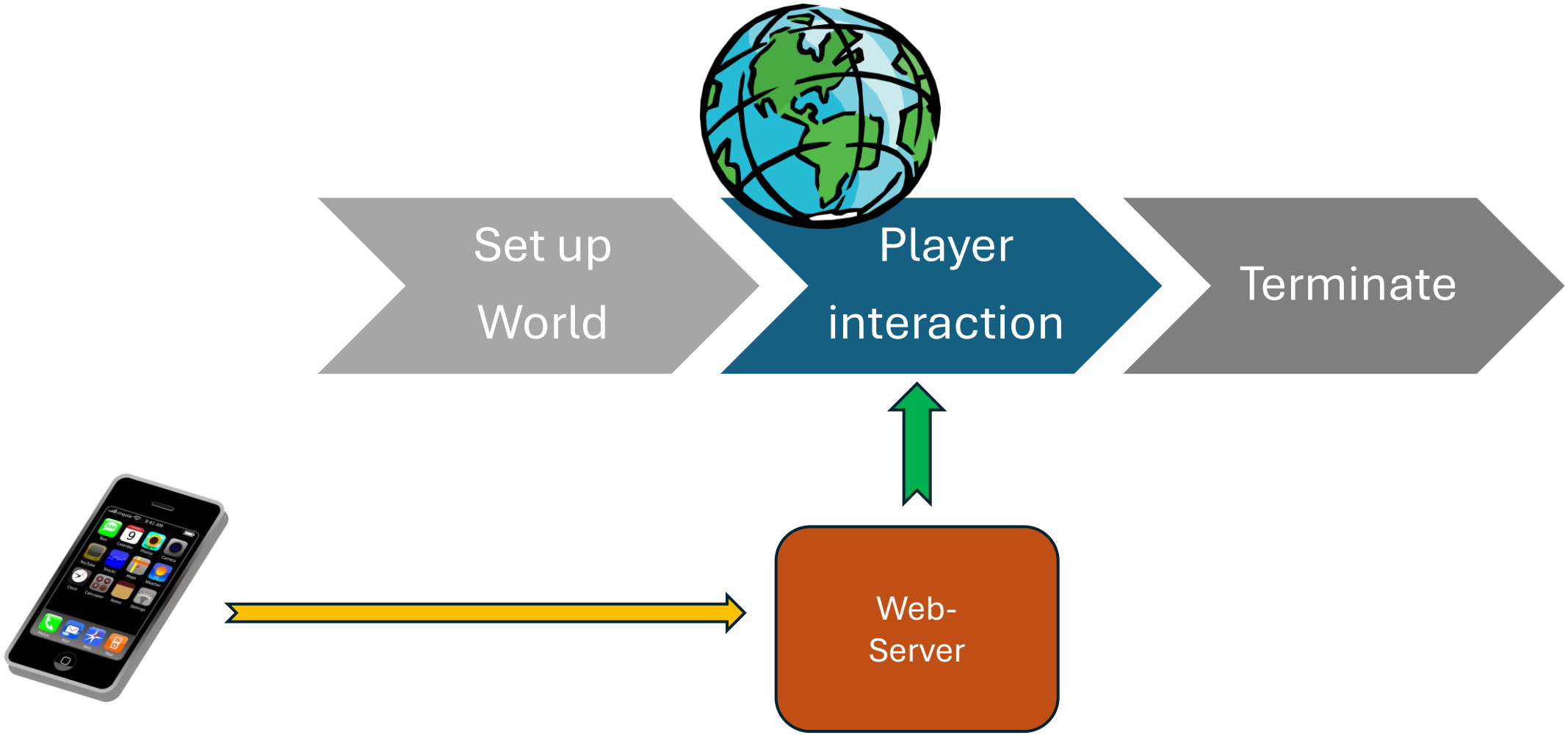

```
REXX-Regina_3.9.5(MT) 5.00 25 Jun 2022
```



Compatibility experiences FASTCGI & PIPE

Function	REGINA	OOREXX	CREXX
a.=	yes	no	no
Stems	yes	yes	(Yes)
Poolid()	yes	no	Name spaces
Sockets	32bit 64 bit as of SYMPOSIUM 2025	32 and 64 bit	planned
SELECT	Yes	Yes	planned

Application Scenario: Game
FASTCGI & PIPE



More...

https://github.com/mikebeer/fastcgi

https://github.com/mikebeer/pipe


Thank you