Unix et Programmation Shell

Philippe Langevin

département d'informatique
UFR sciences et technique
université du sud Toulon Var

Automne 2013

document

brouillon en révision

- site du cours :
  http://langevin.univ-tln.fr/cours/UPS/upsh.html
- localisation du fichier :
  http://langevin.univ-tln.fr/cours/UPS/doc/pipe.pdf

dernières modifications

```
      perm.tex 2016-09-30 09:41:54.766553521 +0200
      file.tex 2016-09-30 09:19:02.810595120 +0200
      bash.tex 2016-09-15 12:09:09.887948313 +0200
      term.tex 2016-09-14 18:50:05.124091515 +0200
      upsh.tex 2015-10-25 18:09:36.027434338 +0100
      proc.tex 2015-10-20 22:09:35.450391618 +0200
       man.tex 2015-09-18 20:13:44.822492893 +0200
     shell.tex 2015-09-10 19:31:04.581529236 +0200
  prologue.tex 2015-09-07 09:06:31.773157847 +0200
     tools.tex 2015-07-11 09:04:38.890915266 +0200
      pipe.tex 2014-10-02 19:10:22.426127326 +0200
    direct.tex 2014-10-02 07:49:17.162784238 +0200
   syntaxe.tex 2014-10-01 23:52:29.859357485 +0200
      part.tex 2014-10-01 23:52:29.372363438 +0200
```

8 - pipeline

traitement des commandes

Le prompt du shell matérialise une ligne de commande invitant l'utilisateur à soumettre une instruction : une chaine de caractères analysée par le shell pour lancer des commandes.

- interne : traitée dans le processus.
- externe : `clone` + `execve`.
- complexe : `clone`

trace des commandes externes

```
1 #!/bin/bash
2 /bin/echo hello
3 echo HELLO
4 /bin/echo allo
```

echos.sh

```
echos.out:echos.sh
 strace -f -e trace=execve,clone,waitpid,write\
          -o echos.str -e signal=SIGCHLD ./
   echos.sh

 sed -n '/echos.out/,/HERE/p' makefile > echos.
   out
```

echos.make

echos.strace

```
19330 execve("./echos.sh", ["./echos.sh"], [/* 54
    vars */]) = 0
19330 clone(child_stack=0, flags=
    CLONE_CHILD_CLEARTID|CLONE_CHILD_SETTID|SIGCHLD,
     child_tidptr=0x7fef183379d0) = 19331
19331 execve("/bin/echo", ["/bin/echo", "hello"],
    [/* 53 vars */]) = 0
19331 write(1, "hello\n", 6)                    = 6
19330 --- SIGCHLD (Child exited) @ 0 (0) ---
19330 write(1, "HELLO\n", 6)                    = 6
19330 clone(child_stack=0, flags=
    CLONE_CHILD_CLEARTID|CLONE_CHILD_SETTID|SIGCHLD,
     child_tidptr=0x7fef183379d0) = 19332
19332 execve("/bin/echo", ["/bin/echo", "allo"],
    [/* 53 vars */]) = 0
19332 write(1, "allo\n", 5)                     = 5
```

pipeline

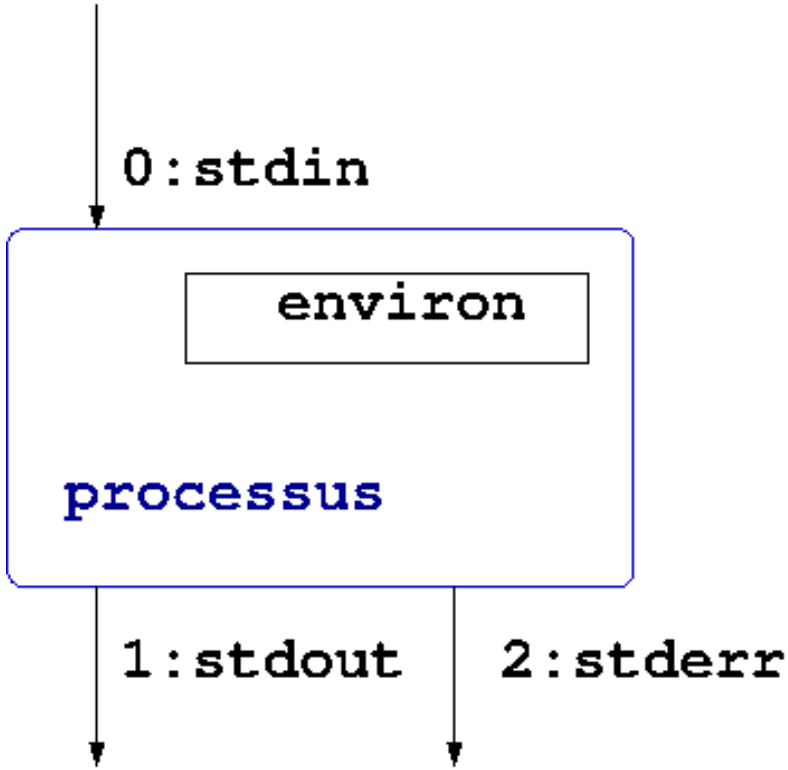
Le pipeline permet de chaîner des processus de sorte que la sortie d'un processus (stdout) alimente directement l'entrée (stdin) du suivant.

```
1 ::    ps -eoppid | sort  |  uniq  -c | sort  -gr
2           |  head -3 | sed 's/[ ]\+/:/g'
3              |  cut  -d':'  -f3 | xargs  ps -p
```

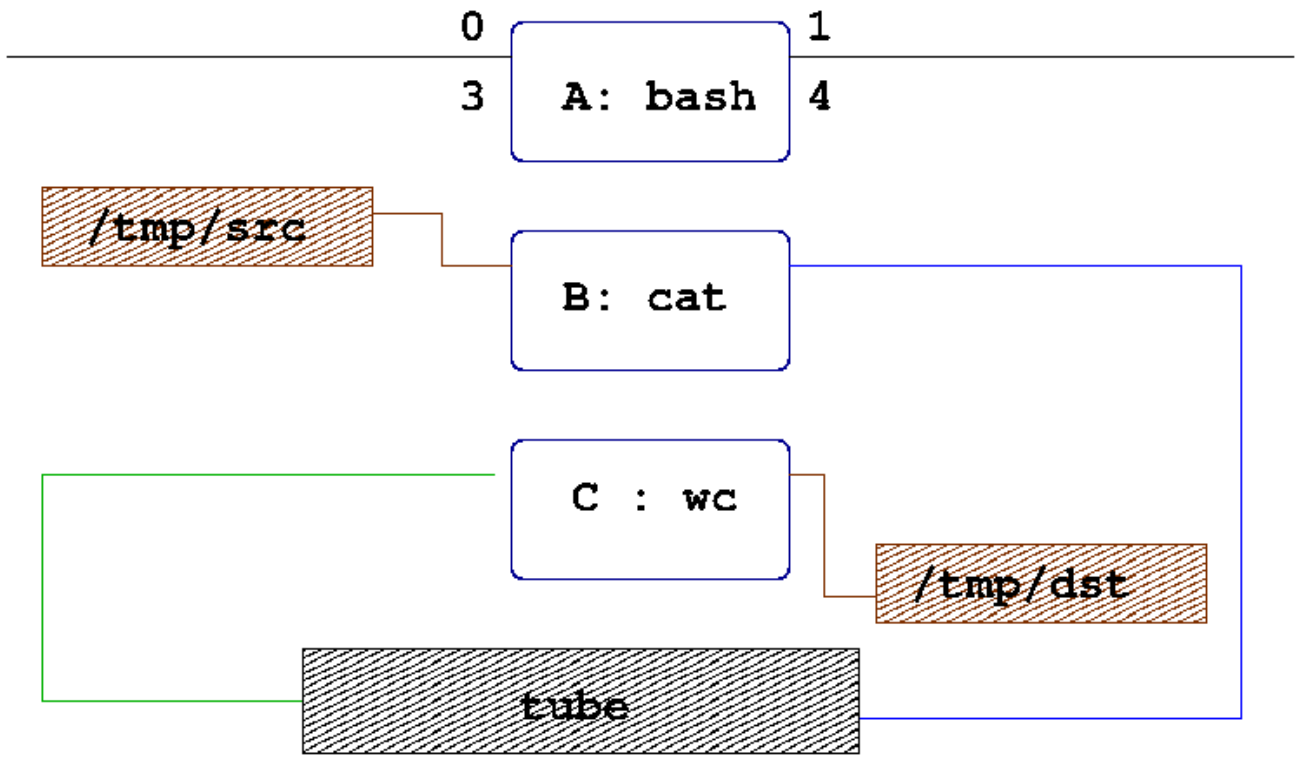
La mise en oeuvre d'un pipeline s'appuie sur les appels système :

- `fork` / `clone`
- `pipe`, `dup2`
- `execve`

processus



cat </tmp/src | wc -l > /tmp/dst



bash pipeline

```
1 echo hello > /tmp/src
2 call =open,close,pipe,dup2,execve,clone,fork
3 strace -o pipe.out -f -e trace=$call -p $$ &
4 sleep 1
5 cat </tmp/src | wc -l >/tmp/dst
```

pipe.sh

```
./pipe.sh > pipe.out
Process 2291 attached
Process 2298 attached
Process 2299 attached
./pipefilter.sh pipe.out
```

trace brute

```
5377  --- SIGCHLD {si_signo=SIGCHLD, si_code=
    CLD_EXITED, si_pid=5379, si_uid=1000,
    si_status=0, si_utime=0, si_stime=0} ---
5377  pipe([3, 4])                          = 0
5377  clone(child_stack=0, flags=
    CLONE_CHILD_CLEARTID|CLONE_CHILD_SETTID|
    SIGCHLD, child_tidptr=0xb778a728) = 5384
5384  close(255)                            = 0
5377  close(4)                              = 0
5377  close(4)                              = -1
    EBADF (Bad file descriptor)
5377  clone( <unfinished ...>
5384  close(3)                              = 0
5384  dup2(4, 1 <unfinished ...>
5377  <... clone resumed> child_stack=0, flags=
```

filtre

```
1  #!/bin/bash
2
3  grep -Ev '(locale|cache|BAD|\-1|255|exited|libc)' pipe.out \
4      | sed -r 's/(clone).*=(.*)/\1 := \2/' \
5      | uniq | tr -d '[A-Z]' | cut -c1-44 > mypipe.out
6
7  set $(grep -oE '^[0-9]+' mypipe.out | sort | uniq )
8  sed -i -e "s/$1/& (A)/g" \
9          -e "s/$2/& (B)/g" \
10         -e "s/$3/& (C)/g" mypipe.out
```

trace filtrée (1/3)

```
5377 (A)  ---  {si_signo=, si_code=_, si_pid=537
5377 (A)  pipe(3, 4)                          = 0
5377 (A)  clone :=  5384 (B)
5377 (A)  close(4)                            = 0
5377 (A)  clone( <unfinished ...>
5384 (B)  close(3)                            = 0
5384 (B)  dup2(4, 1 <unfinished ...>
5377 (A)  <... clone :=  5385 (C)
5384 (B)  <... dup2 resumed> )                = 1
5384 (B)  close(4)                            = 0
5377 (A)  close(3)                            = 0
5385 (C)  dup2(3, 0)                          = 0
```

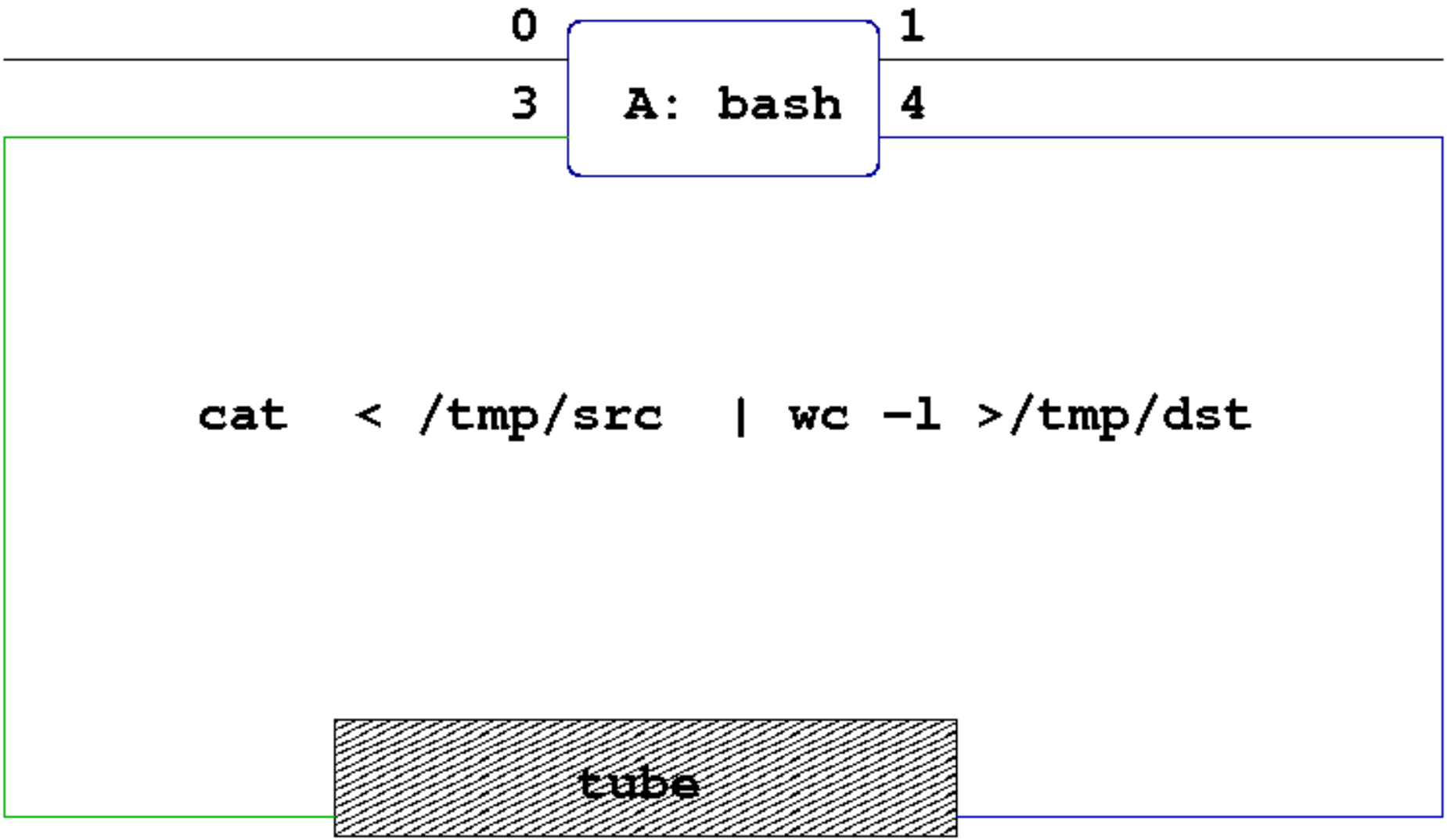
trace filtrée (2/3)

```
5385 (C)  close(3)                                  = 0
5384 (B)  open("/tmp/src", _|_) = 3
5384 (B)  dup2(3, 0)                                = 0
5384 (B)  close(3)                                  = 0
5384 (B)  execve("/usr/bin/cat", "cat", /* 55 va
5385 (C)  open("/tmp/dst", _|_|_|_, 0666) = 3
5385 (C)  dup2(3, 1)                                = 1
5385 (C)  close(3)                                  = 0
5384 (B)  <... execve resumed> )                    = 0
5385 (C)  execve("/usr/bin/wc", "wc", "-l", /* 5
5385 (C)  <... execve resumed> )                    = 0
5384 (B)  <... open resumed> )                      = 3
5384 (B)  close(3)                                  = 0
```

trace filtrée (2/3)

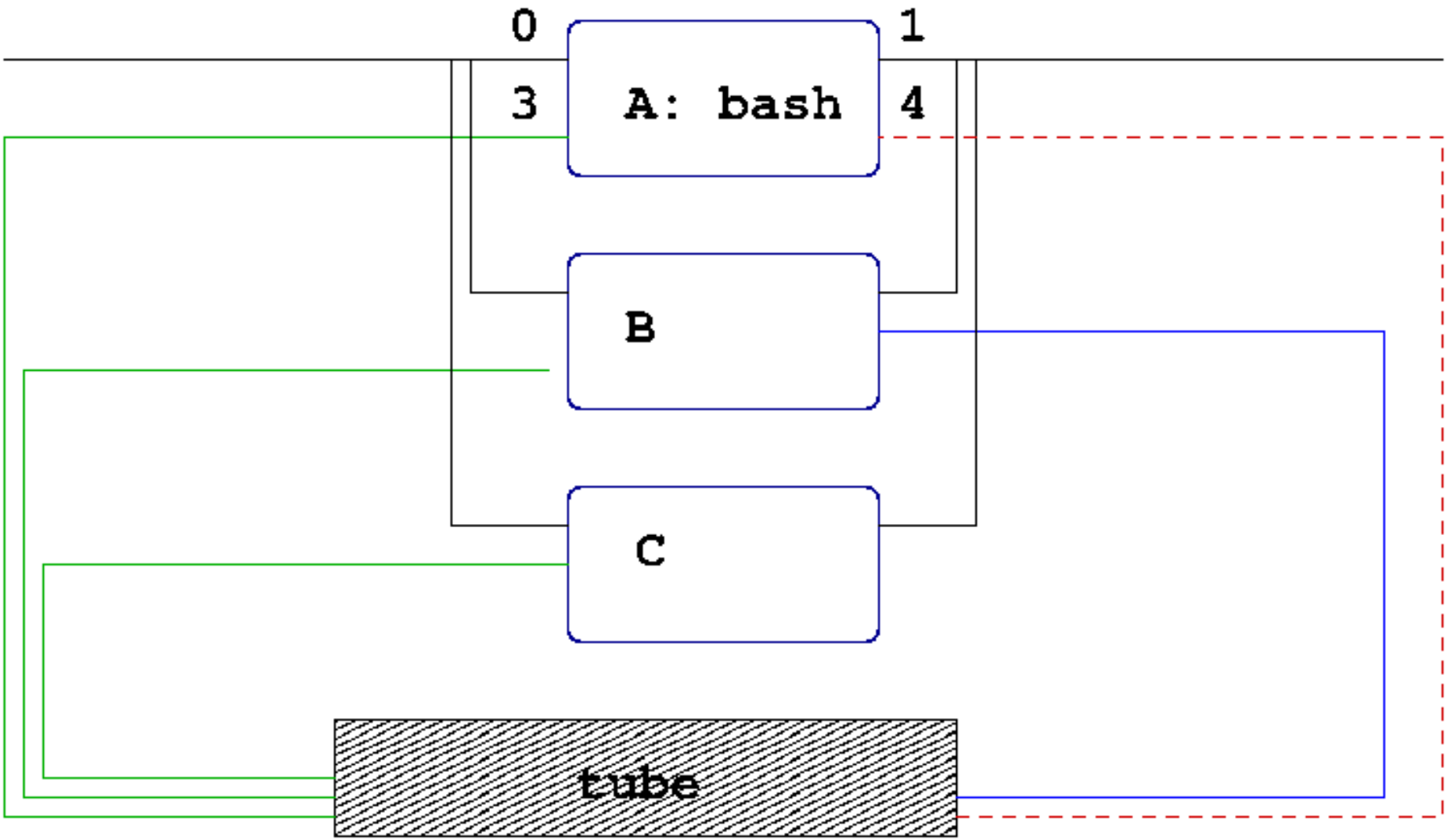
```
5385 (C)   <... open resumed> )                    = 3
5384 (B)   <... open resumed> )                    = 3
5385 (C)   close(3)                                = 0
5384 (B)   close(3)                                = 0
5385 (C)   close(3)                                = 0
5384 (B)   close(3)                                = 0
5385 (C)   close(3)                                = 0
5384 (B)   close(0)                                = 0
5384 (B)   close(1 <unfinished ...>
5385 (C)   close(3 <unfinished ...>
5384 (B)   <... close resumed> )                   = 0
5385 (C)   <... close resumed> )                   = 0
5384 (B)   close(2)                                = 0
5385 (C)   close(3)                                = 0
5385 (C)   close(0)                                = 0
5385 (C)   close(1)                                = 0
```

A : pipe(3,4)



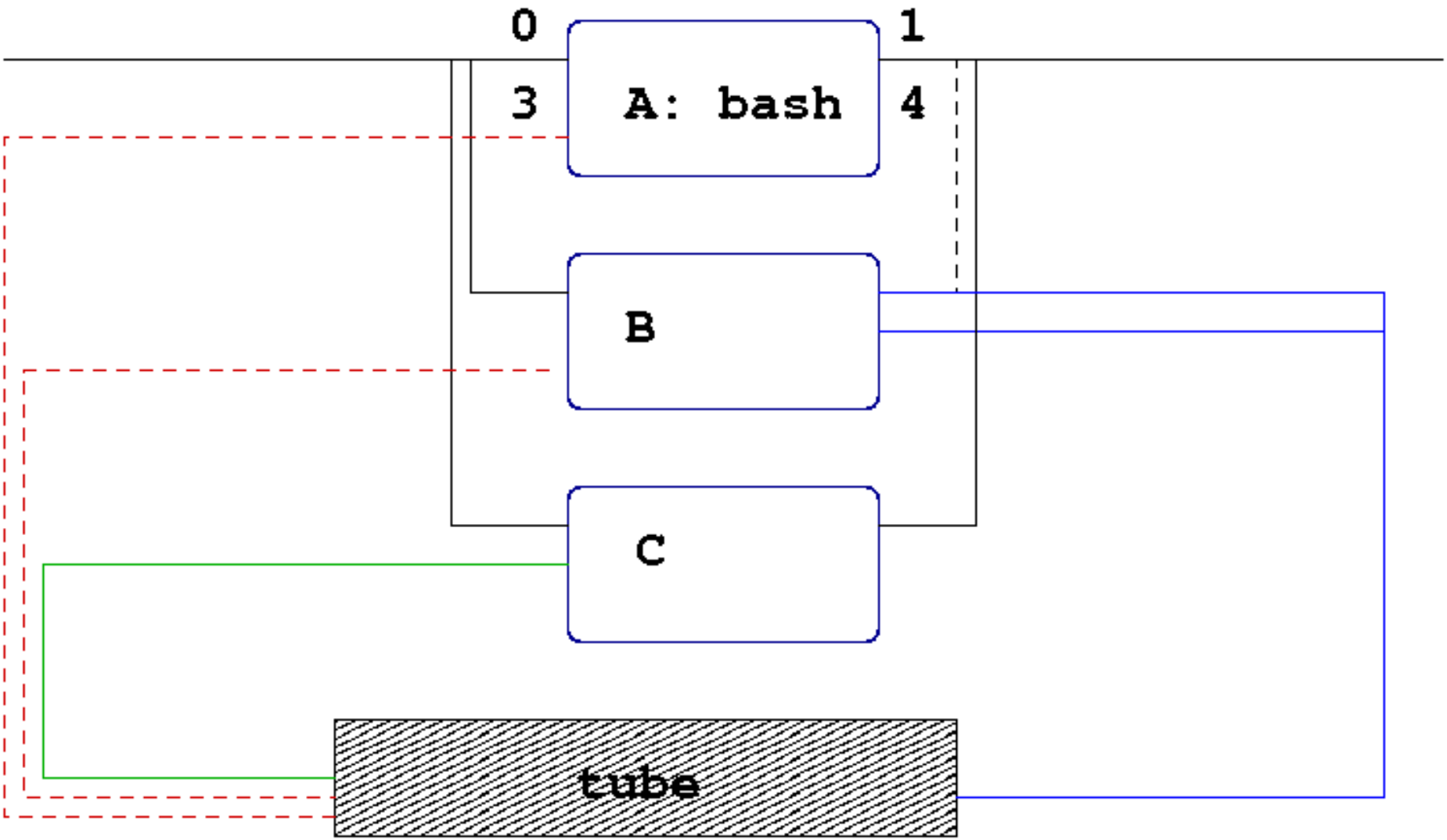
A : clone,close(4), clone

A : clone, close(4), clone



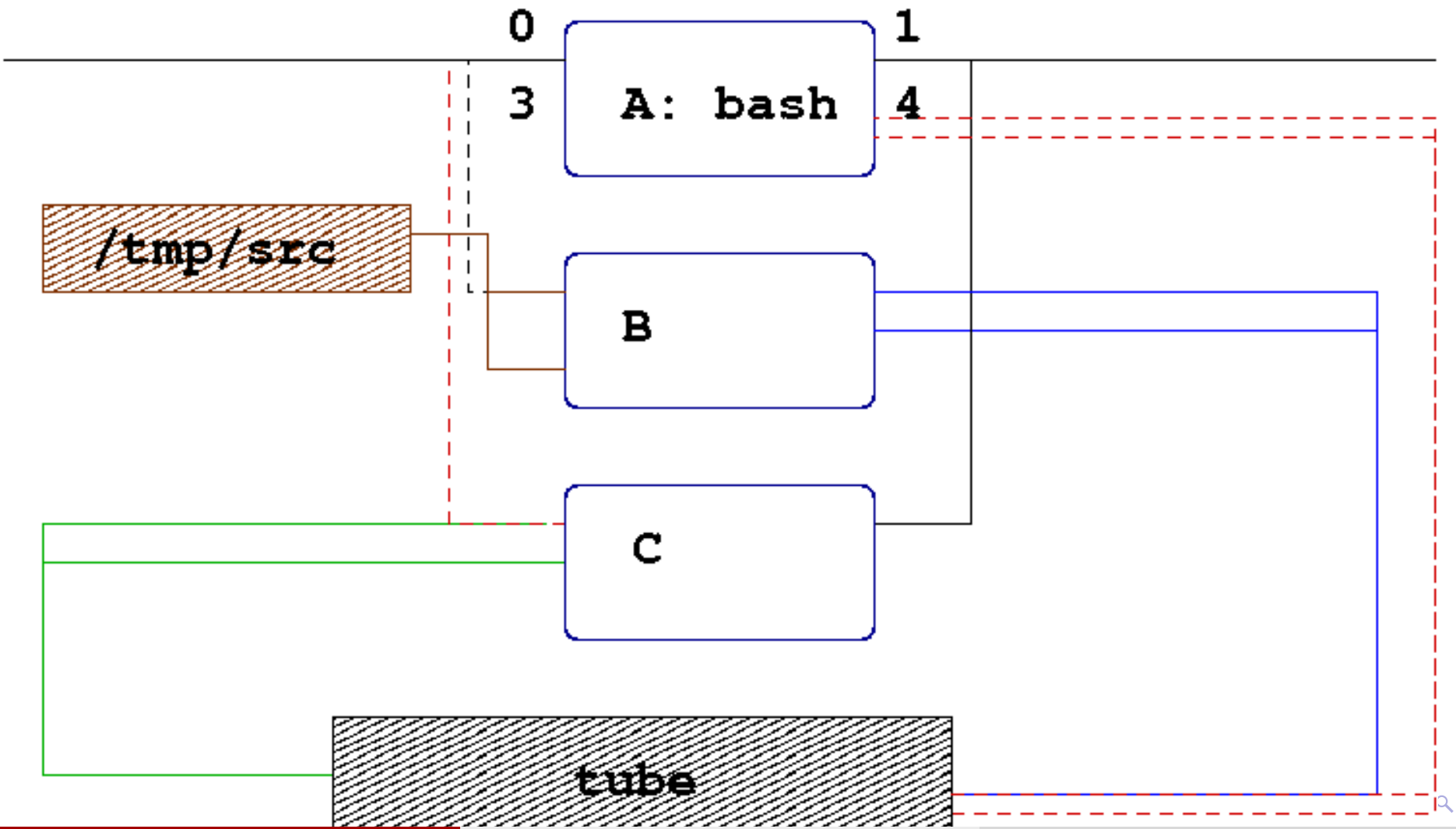
B : close(3), dup2(4,1), A : close(3)

B : close(3), dup2(4,1), A : close(3)

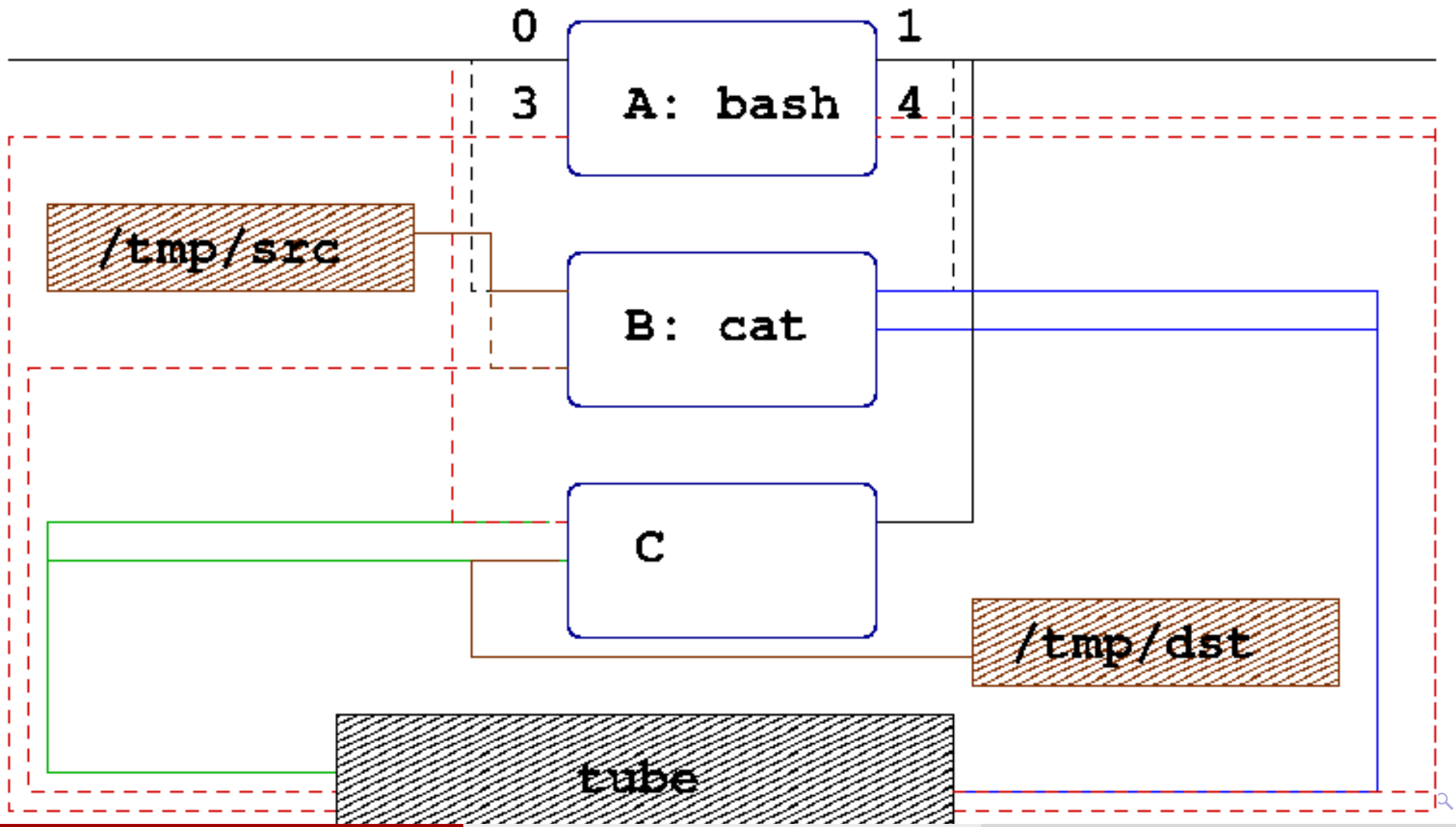


C : dup2(3,0), B : open(/tmp/src) = 3, B : dup2(3,0)

C : dup2(3,0), B : open(/tmp/src)=3, B : dup2(3,0)



B : close(3), C : open(/tmp/dst)=3, B : execve(cat)



C : dup2(3,1), close(3), execve(wc)