



- ```

graph LR
 subgraph processus
 fd1[fd[1]] -- pipe --> fd0[fd[0]]
 end

```

```
#include <stdio.h>
#include <memory.h>
#include <unistd.h>

int main(int argc, char ** argv)
{
 char buffer[BUFSIZ+1];

 /* create the pipe */
 int fd[2];
 pipe(fd);

 /* write into the pipe */
 write(fd[1], "Hello World\n", strlen("Hello World\n"));

 /* read the pipe and print the read value */
 read(fd[0], buffer, BUFSIZ);
 printf("%s", buffer);
}
```

## Création d'un pipe dans un processus ayant un fils

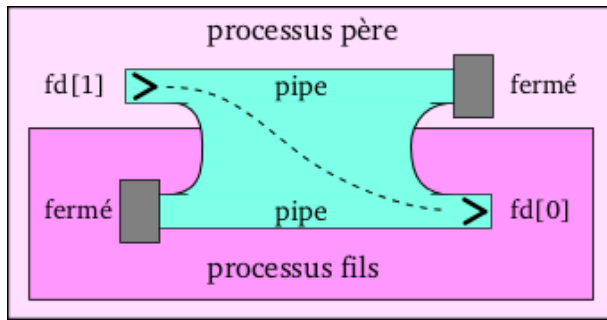
```

graph LR
 subgraph "processus père"
 direction LR
 P1[] --> P2[]
 P2 --> P3[]
 P3 --> P4[]
 P4 --> P5[]
 P5 --> P6[]
 P6 --> P7[]
 P7 --> P8[]
 P8 --> P9[]
 P9 --> P10[]
 P10 --> P11[]
 P11 --> P12[]
 P12 --> P13[]
 P13 --> P14[]
 P14 --> P15[]
 P15 --> P16[]
 P16 --> P17[]
 P17 --> P18[]
 P18 --> P19[]
 P19 --> P20[]
 P20 --> P21[]
 P21 --> P22[]
 P22 --> P23[]
 P23 --> P24[]
 P24 --> P25[]
 P25 --> P26[]
 P26 --> P27[]
 P27 --> P28[]
 P28 --> P29[]
 P29 --> P30[]
 P30 --> P31[]
 P31 --> P32[]
 P32 --> P33[]
 P33 --> P34[]
 P34 --> P35[]
 P35 --> P36[]
 P36 --> P37[]
 P37 --> P38[]
 P38 --> P39[]
 P39 --> P40[]
 P40 --> P41[]
 P41 --> P42[]
 P42 --> P43[]
 P43 --> P44[]
 P44 --> P45[]
 P45 --> P46[]
 P46 --> P47[]
 P47 --> P48[]
 P48 --> P49[]
 P49 --> P50[]
 P50 --> P51[]
 P51 --> P52[]
 P52 --> P53[]
 P53 --> P54[]
 P54 --> P55[]
 P55 --> P56[]
 P56 --> P57[]
 P57 --> P58[]
 P58 --> P59[]
 P59 --> P60[]
 P60 --> P61[]
 P61 --> P62[]
 P62 --> P63[]
 P63 --> P64[]
 P64 --> P65[]
 P65 --> P66[]
 P66 --> P67[]
 P67 --> P68[]
 P68 --> P69[]
 P69 --> P70[]
 P70 --> P71[]
 P71 --> P72[]
 P72 --> P73[]
 P73 --> P74[]
 P74 --> P75[]
 P75 --> P76[]
 P76 --> P77[]
 P77 --> P78[]
 P78 --> P79[]
 P79 --> P80[]
 P80 --> P81[]
 P81 --> P82[]
 P82 --> P83[]
 P83 --> P84[]
 P84 --> P85[]
 P85 --> P86[]
 P86 --> P87[]
 P87 --> P88[]
 P88 --> P89[]
 P89 --> P90[]
 P90 --> P91[]
 P91 --> P92[]
 P92 --> P93[]
 P93 --> P94[]
 P94 --> P95[]
 P95 --> P96[]
 P96 --> P97[]
 P97 --> P98[]
 P98 --> P99[]
 P99 --> P100[]
 P100 --> P101[]
 P101 --> P102[]
 P102 --> P103[]
 P103 --> P104[]
 P104 --> P105[]
 P105 --> P106[]
 P106 --> P107[]
 P107 --> P108[]
 P108 --> P109[]
 P109 --> P110[]
 P110 --> P111[]
 P111 --> P112[]
 P112 --> P113[]
 P113 --> P114[]
 P114 --> P115[]
 P115 --> P116[]
 P116 --> P117[]
 P117 --> P118[]
 P118 --> P119[]
 P119 --> P120[]
 P120 --> P121[]
 P121 --> P122[]
 P122 --> P123[]
 P123 --> P124[]
 P124 --> P125[]
 P125 --> P126[]
 P126 --> P127[]
 P127 --> P128[]
 P128 --> P129[]
 P129 --> P130[]
 P130 --> P131[]
 P131 --> P132[]
 P132 --> P133[]
 P133 --> P134[]
 P134 --> P135[]
 P135 --> P136[]
 P136 --> P137[]
 P137 --> P138[]
 P138 --> P139[]
 P139 --> P140[]
 P140 --> P141[]
 P141 --> P142[]
 P142 --> P143[]
 P143 --> P144[]
 P144 --> P145[]
 P145 --> P146[]
 P146 --> P147[]
 P147 --> P148[]
 P148 --> P149[]
 P149 --> P150[]
 P150 --> P151[]
 P151 --> P152[]
 P152 --> P153[]
 P153 --> P154[]
 P154 --> P155[]
 P155 --> P156[]
 P156 --> P157[]
 P157 --> P158[]
 P158 --> P159[]
 P159 --> P160[]
 P160 --> P161[]
 P161 --> P162[]
 P162 --> P163[]
 P163 --> P164[]
 P164 --> P165[]
 P165 --> P166[]
 P166 --> P167[]
 P167 --> P168[]
 P168 --> P169[]
 P169 --> P170[]
 P170 --> P171[]
 P171 --> P172[]
 P172 --> P173[]
 P173 --> P174[]
 P174 --> P175[]
 P175 --> P176[]
 P176 --> P177[]
 P177 --> P178[]
 P178 --> P179[]
 P179 --> P180[]
 P180 --> P181[]
 P181 --> P182[]
 P182 --> P183[]
 P183 --> P184[]
 P184 --> P185[]
 P185 --> P186[]
 P186 --> P187[]
 P187 --> P188[]
 P188 --> P189[]
 P189 --> P190[]
 P190 --> P191[]
 P191 --> P192[]
 P192 --> P193[]
 P193 --> P194[]
 P194 --> P195[]
 P195 --> P196[]
 P196 --> P197[]
 P197 --> P198[]
 P198 --> P199[]
 P199 --> P200[]
 P200 --> P201[]
 P201 --> P202[]
 P202 --> P203[]
 P203 --> P204[]
 P204 --> P205[]
 P205 --> P206[]
 P206 --> P207[]
 P207 --> P208[]
 P208 --> P209[]
 P209 --> P210[]
 P210 --> P211[]
 P211 --> P212[]
 P212 --> P213[]
 P213 --> P214[]
 P214 --> P215[]
 P215 --> P216[]
 P216 --> P217[]
 P217 --> P218[]
 P218 --> P219[]
 P219 --> P220[]
 P220 --> P221[]
 P221 --> P222[]
 P222 --> P223[]
 P223 --> P224[]
 P224 --> P225[]
 P225 --> P226[]
 P226 --> P227[]
 P227 --> P228[]
 P228 --> P229[]
 P229 --> P230[]
 P230 --> P231[]
 P231 --> P232[]
 P232 --> P233[]
 P233 --> P234[]
 P234 --> P235[]
 P235 --> P236[]
 P236 --> P237[]
 P237 --> P238[]
 P238 --> P239[]
 P239 --> P240[]
 P240 --> P241[]
 P241 --> P242[]
 P242 --> P243[]
 P243 --> P244[]
 P244 --> P245[]
 P245 --> P246[]
 P246 --> P247[]
 P247 --> P248[]
 P248 --> P249[]
 P249 --> P250[]
 P250 --> P251[]
 P251 --> P252[]
 P252 --> P253[]
 P253 --> P254[]
 P254 --> P255[]
 P255 --> P256[]
 P256 --> P257[]
 P257 --> P258[]
 P258 --> P259[]
 P259 --> P260[]
 P260 --> P261[]
 P261 --> P262[]
 P262 --> P263[]
 P263 --> P264[]
 P264 --> P265[]
 P265 --> P266[]
 P266 --> P267[]
 P267 --> P268[]
 P268 --> P269[]

```

## Découvrir

Pour être certain de qui va écrire et qui va lire dans le pipe, il faut que les processus ferment les extrémités qu'ils n'utilisent pas.



De cette façon le processus père peut être certain que s'il écrit dans le pipe ("fd[1]"), le fils va recevoir l'information en lecture ("fd[0]").

Concretement voici comment faire en C :

```
#include <stdio.h>
#include <memory.h>
#include <unistd.h>

int main(int argc, char ** argv)
{
 /* create the pipe */
 int pfd[2];
 if (pipe(pfd) == -1)
 {
 printf("pipe failed\n");
 return 1;
 }

 /* create the child */
 int pid;
 if ((pid = fork()) < 0)
 {
 printf("fork failed\n");
 return 2;
 }

 if (pid == 0)
 {
 /* child */
 char buffer[BUFSIZ];

 close(pfd[1]); /* close write side */

 /* read some data and print the result on screen */
 while (read(pfd[0], buffer, BUFSIZ) != 0)
 printf("child reads %s", buffer);

 close(pfd[0]); /* close the pipe */
 }
 else
 {
 /* parent */
 char buffer[BUFSIZ];

 close(pfd[0]); /* close read side */

 /* send some data into the pipe */
 strcpy(buffer, "HelloWorld\n");
 write(pfd[1], buffer, strlen(buffer)+1);

 close(pfd[1]); /* close the pipe */
 }

 return 0;
}
```



## Drive a vintage car in Tuscany

Travel the beautiful Tuscany on board an original the 60s

Drive the Vintage



## Tableau blanc virtuel en ligne

Facilitez l'organisation de vos brainstormings col grâce à nos outils puissants.

Lucidspark

S'i

## stdin, stdout, stderr

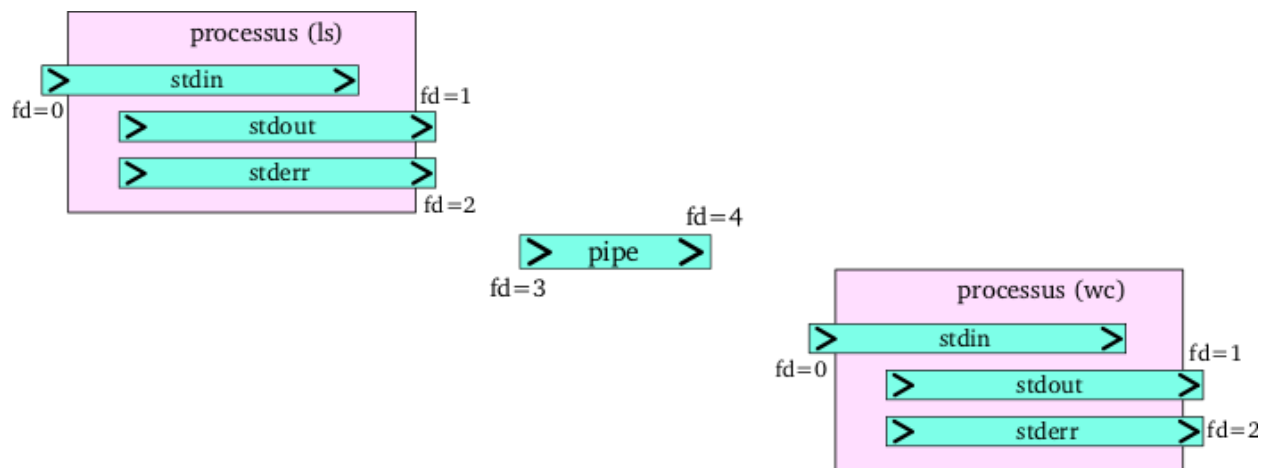
Ces trois pipes sont par défaut créés dans chaque processus. Le premier, stdin, est branché par défaut sur l'entrée clavier tandis que stdout et stderr sont eux branchés sur la sortie écran. Des descripteurs de fichiers par défaut leur sont associés : 0 pour stdin, 1 pour stdout, et 2 pour stderr.

Maintenant que l'on connaît le principe de communication par pipe, on est tenté de connecter ces pipes entre eux. Par exemple essayons de connecter stdout d'un premier processus avec le stdin d'un second. Cette opération est effectuée par le shell à chaque fois que deux commandes séparées par un pipe sont exécutées, par exemple pour relier la sortie stdout de la commande ls avec l'entrée stdin de la commande wc, il faut taper ceci dans un terminal : "ls | wc".

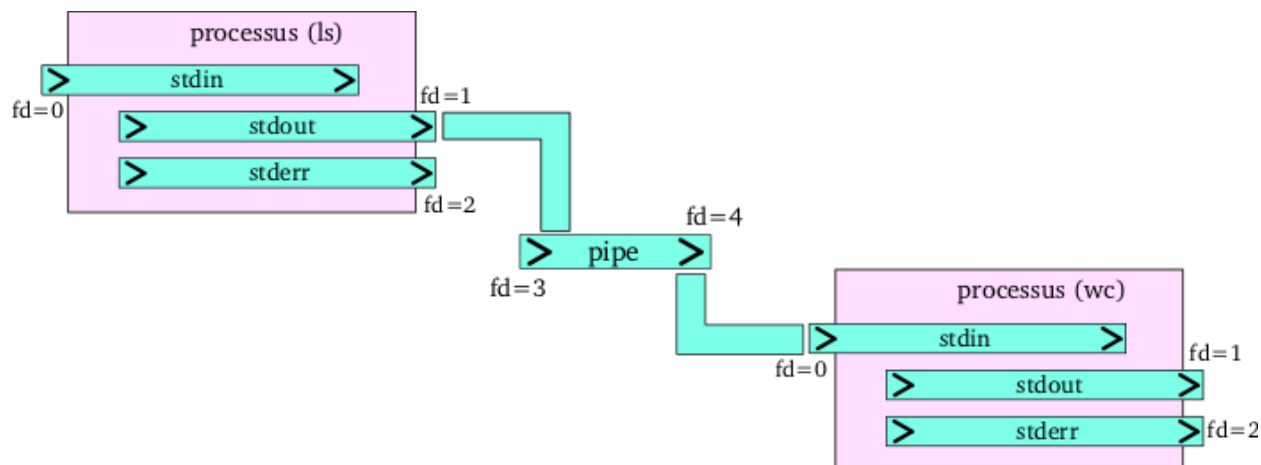
Pour réaliser ceci dans un programme en langage C, il faut procéder en plusieurs étapes.

Il faut commencer par créer un pipe vide : "fd=3" en écriture et "fd=4" en lecture. On utilise donc la fonction "pipe(fd)" comme vue ci dessus.

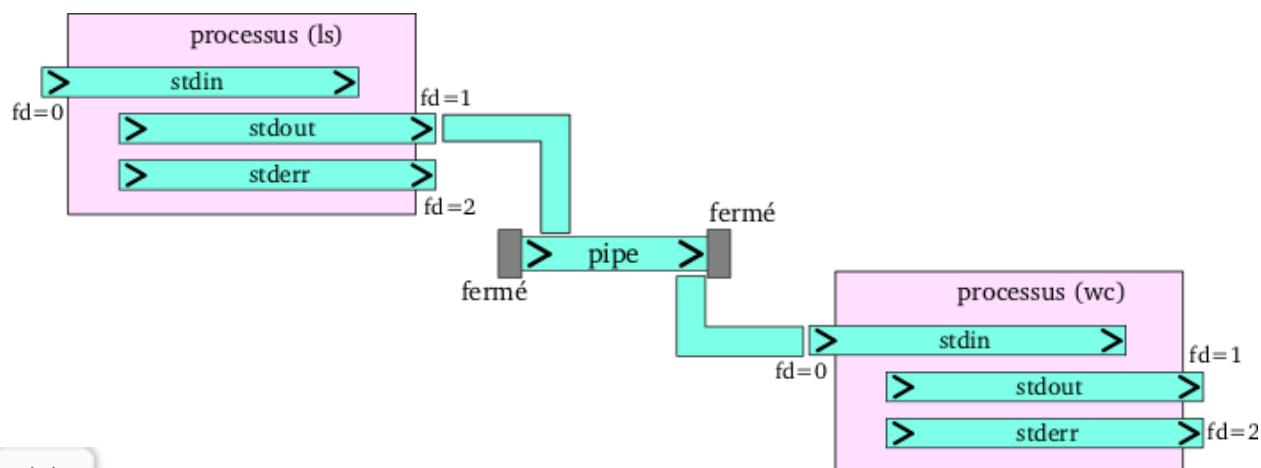




Ensuite il faut que stdout du premier processus (fd1) soit connecté à l'entrée de notre pipe (fd3) et que la sortie de notre pipe (fd4) soit connecté à stdin de notre second processus (fd0). On appellera deux fois la fonction "dup2(param1, param2)" pour connecter fd1 à fd3 et fd4 à fd0. "dup2" prend en argument deux paramètres, ce sont des descripteurs de fichiers : param1 vaudra fd3 et param2 vaudra fd1 car on veut que fd3 soit assimilé (ou connecté) à fd1. Pour le second processus, param1 vaudra fd4 et param2 vaudra fd0.



Finalement il faut fermer les extrémités de notre pipe pour éviter les comportements étranges, c'est un peu le même problème que dans la section précédente. On utilisera la commande "close(fd)" pour fermer les extrémités de notre pipe.



```
/* create the pipe */
int pfd[2];
if (pipe(pfd) == -1)
{
 printf("pipe failed\n");
 return 1;
}

/* create the child */
int pid;
if ((pid = fork()) < 0)
{
 printf("fork failed\n");
 return 2;
}

if (pid == 0)
{
 /* child */
 close(pfd[1]); /* close the unused write side */
 dup2(pfd[0], 0); /* connect the read side with stdin */
 close(pfd[0]); /* close the read side */
 /* execute the process (wc command) */
 execlp("wc", "wc", (char *) 0);
 printf("wc failed"); /* if execlp returns, it's an error */
 return 3;
}
else
{
 /* parent */
 close(pfd[0]); /* close the unused read side */
 dup2(pfd[1], 1); /* connect the write side with stdout */
 close(pfd[1]); /* close the write side */
 /* execute the process (ls command) */
 execlp("ls", "ls", (char *) 0);
 printf("ls failed"); /* if execlp returns, it's an error */
 return 4;
}
return 0;
}
```

Ici on découvre l'utilisation d'une nouvelle fonction "execlp(...)" qui permet de remplacer le processus en cours par l'exécution d'une commande : dans notre cas la commande "wc" est appelée dans le processus fils et la commande "ls" est appelée dans le processus père. Cette fonction ne retourne rien sauf si une erreur se produit à l'exécution de la commande.

## Conclusion

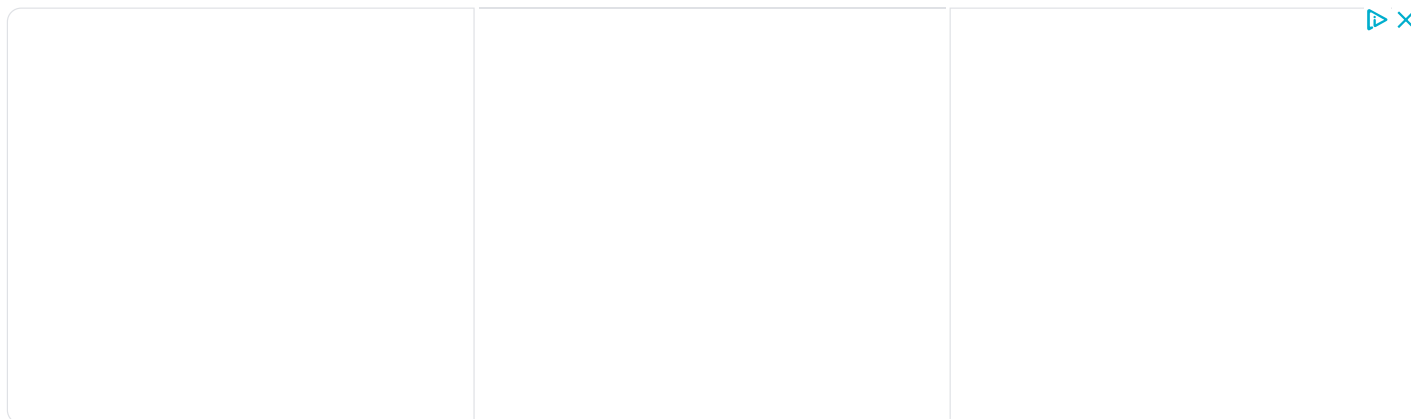
On a vu dans cet article comment créer des pipes mais on peut se demander à quoi peuvent ils servir. Il y a certainement une infinité d'applications associées à ce mécanisme mais voici l'idée que j'avais derrière la tête et qui m'a poussé à étudier les pipes : créer une interface utilisateur (GUI) permettant de contrôler un programme en ligne de commande (qui utilise donc stdin, stdout et stderr pour communiquer avec l'extérieur). En effet, si on peut contrôler l'entrée (stdin) et la sortie (stdout, stderr) d'un programme en l'encapsulant dans un autre programme père, on peut tout faire avec !



## Drive a vintage car in Tuscany

Travel the beautiful Tuscany on board an original  
the 60s





## Babyfoot Vintage

### Sources

---

- <http://www.cryptnet.net/fdp/gtk/exe...> [<http://www.cryptnet.net/fdp/gtk/exec.html>]
- <http://pandonia.canberra.edu.au/OS/...> [[http://pandonia.canberra.edu.au/OS/l9\\_1.html](http://pandonia.canberra.edu.au/OS/l9_1.html)]
- <http://www.gidforums.com/t-3369.html> [<http://www.gidforums.com/t-3369.html>]
- <http://mkssoftware.com/docs/man3/ex...> [<http://mkssoftware.com/docs/man3/execl.3.asp>]
- <http://www.opengroup.org/onlinepubs...> [<http://www.opengroup.org/onlinepubs/009695399/functions/dup.html>]