TP 5

SYSTEMES D'EXPLOITATION

SCHEDULE

- 1er Semaine: 1 serveur – 1 client
- **2éme Semaine: 1 serveur – n client**

INFORMATIONS GÉNÉRALES

- Makefile pour multiple exécutables
- Multiple client avec fork()
- Processus Zombie / Processus Orphelin
- Références au cours :
  - 9. Processus et communication inter-processusFile

MAKEFILE

- Serveur.c (main())
- Fonctions.c

- Le executable **serveur** sera généré en compilant deux fichiers source: serveur.c et fonctions.c

```
serveur: serveur.c fonctions.c
  gcc -o serveur serveur.c fonctions.c

$make serveur
```

MAKEFILE

- Serveur.c (main())
- Fonctions.c
- Client.c (main())

- Le executable **serveur** sera généré en compilant deux fichiers source: serveur.c et fonctions.c
- Le executable **client** sera généré en compilant deux fichiers source: client.c et fonctions.c

```
client: client.c fonctions.c
  gcc –o client client.c fonctions.c

$make client
```

MAKEFILE

Target **all** will not generate a file

Pseudo target

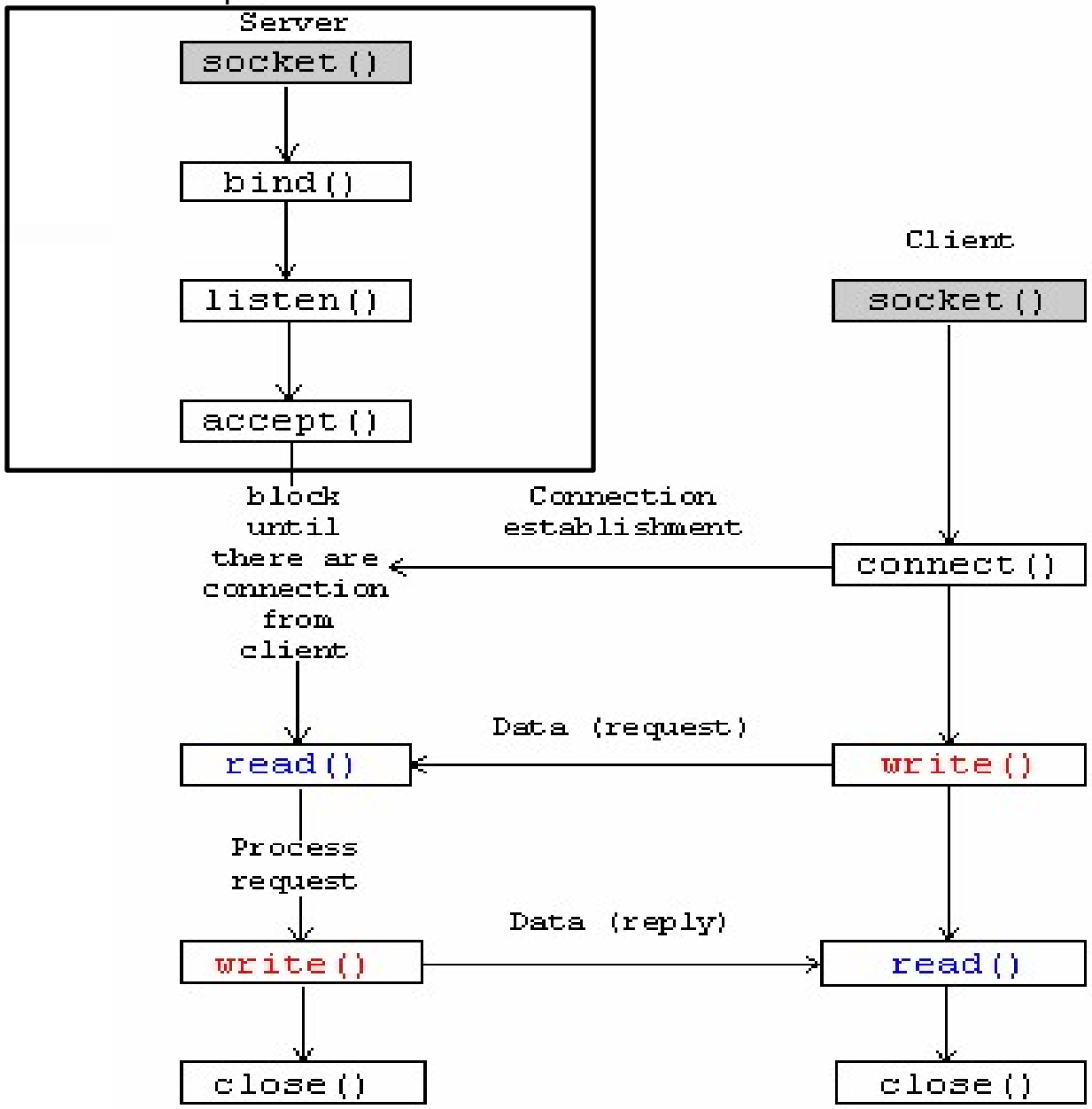
```
.PHONY: all
all: serveur client

serveur: serveur.c fonctions.c
	gcc -o serveur serveur.c fonctions.c

client: client.c fonctions.c
	gcc -o client client.c fonctions.c
```

```
$make
$make all
$make client
$make serveur
```

FORK

- Références au cours:
  - 9. Processus et virtualisation

Processus parent



FORK

```
#include <unistd.h>
pid_t pid = fork();
if (pid > 0) {
  // code du parent
}
else if (pid == 0){
  // code de l'enfant
}
else {
  // erreur
}
```

- Le processus enfant n'est pas une réplique exacte du parent (voir le manuel), notamment:
  - L'enfant a son propre PID et son PPID est égale au PID du parent
  - Pas d'héritage des verrous
  - Pas de signaux en attente

FORK

- Lorsqu'un processus effectue un fork, il doit donc prendre soins d'éviter les zombie en appelant une des fonctions suivantes:
  - `pid_t wait(int *status);`
  - `pid_t waitpid(pid_t pid, int *status, int options);`
- Ces fonctions permettent d'attendre la terminaison d'un enfant pour récupérer son statut. Si un enfant est déjà terminé (I.e. est un zombie), ces fonctions retournent immédiatement.
- Plusieurs macros permettent de tester le statut de retour:
  - `WIFEXITED(status)`: indique si l'enfant c'est terminé normalement
  - `WCOREDUMP(status)`: indique si un *core dump* de l'enfant a été créé.

FORK

- Fonctions utiles:
  - fork()
  - getpid()
  - getppid()
  - wait()
  - waitpid()
  - kill()

CONSEILS - FORK

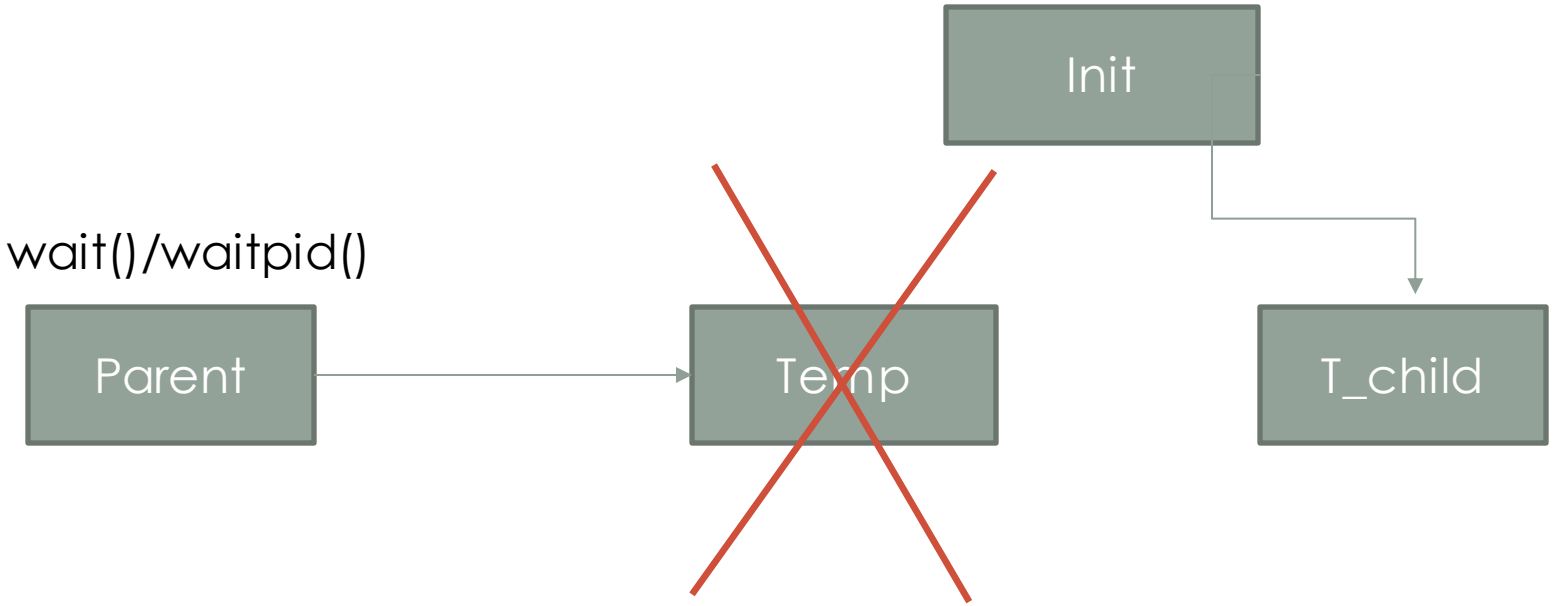
- Il faut éviter des zombies, comment?

SOLUTION

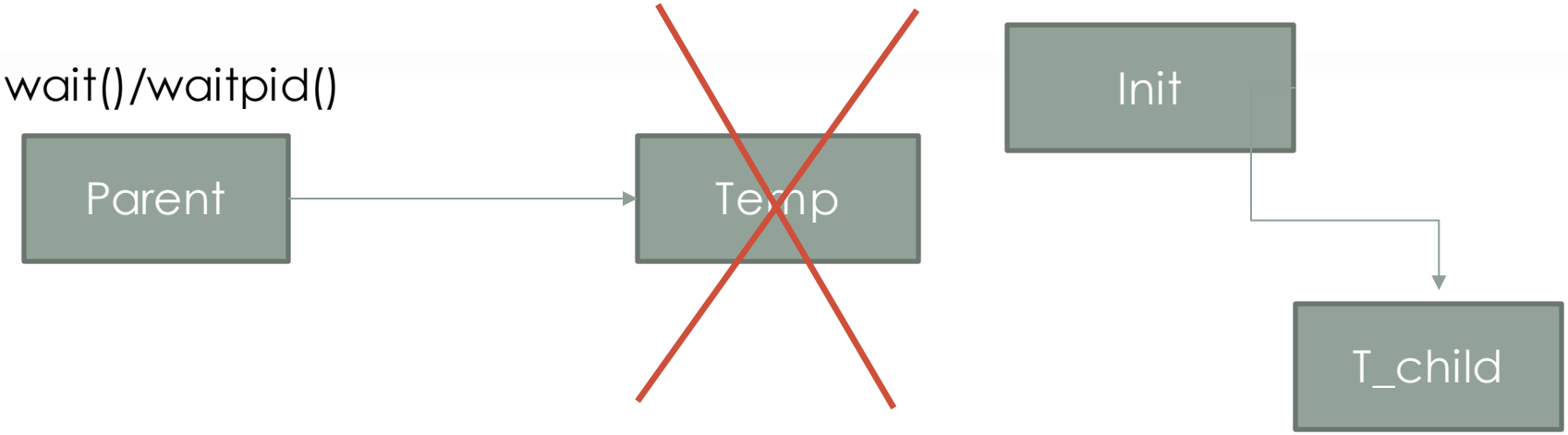
- fork() twice and let init process take care of the orphan

SOLUTION

- Fork() twice and let init process take care of the orphan



SOLUTION




```
accept();
t_pid = fork();
if (t_pid==0){
 pid = fork();
 if(pid==0){
  handleclient();
 }
 else{
  exit();
 }
waitpid(t_pid);
```