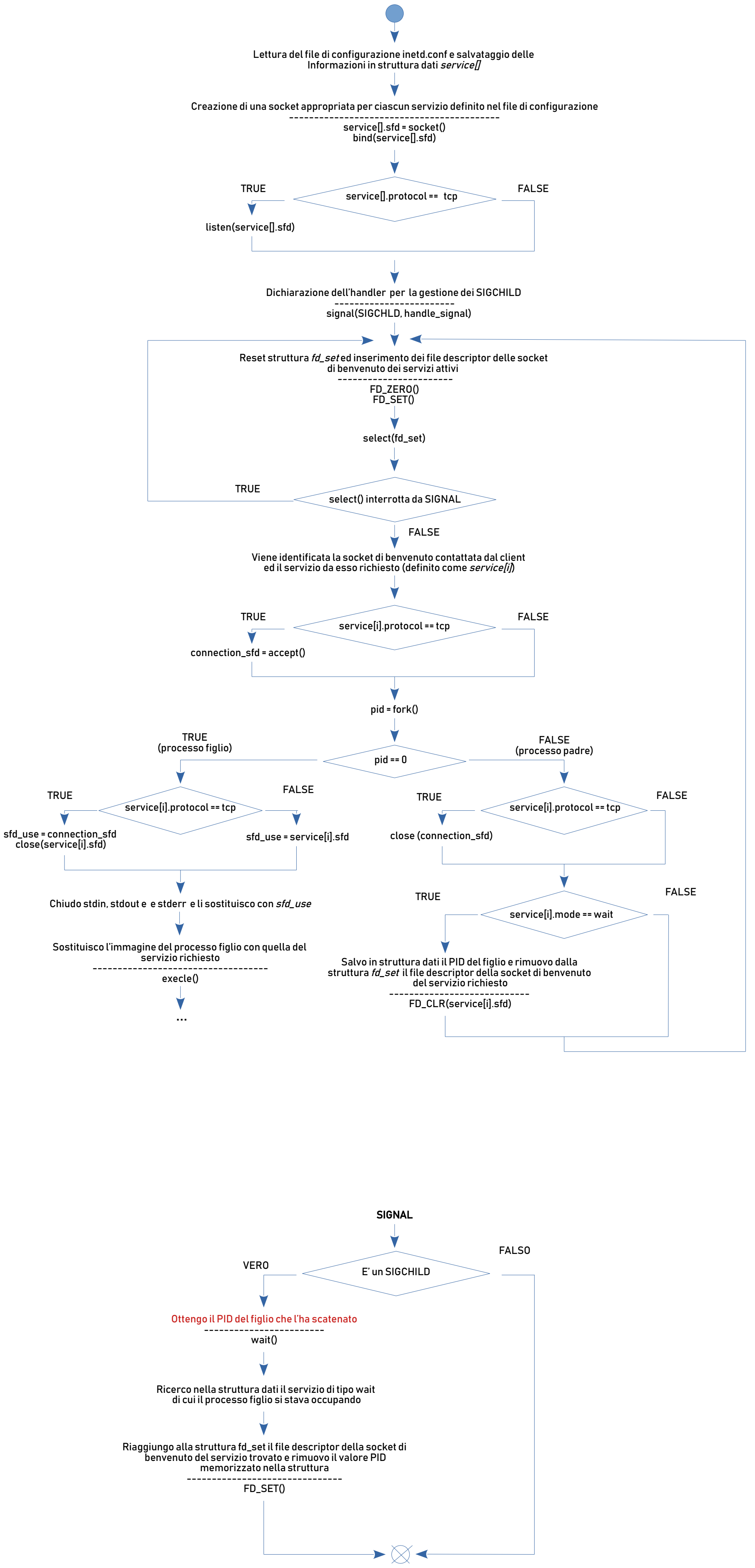
Lettura del file di configurazione inetd.conf e salvataggio delle Informazioni in struttura dati *service[]*
Creazione di una socket appropriata per ciascun servizio definito nel file di configurazione
---
service[].sfd = socket()
bind(service[].sfd)
TRUE
service[].protocol == tcp
FALSE
listen(service[].sfd)
Dichiarazione dell'handler per la gestione dei SIGCHILD
---
signal(SIGCHLD, handle_signal)
Reset struttura *fd_set* ed inserimento dei file descriptor delle socket di benvenuto dei servizi attivi
---
FD_ZERO()
FD_SET()
select(fd_set)
TRUE
select() interrotta da SIGNAL
FALSE
Viene identificata la socket di benvenuto contattata dal client ed il servizio da esso richiesto (definito come *service[i]*)
TRUE
service[i].protocol == tcp
FALSE
connection_sfd = accept()
pid = fork()
TRUE (processo figlio)
pid == 0
FALSE (processo padre)
TRUE
service[i].protocol == tcp
FALSE
sfd_use = connection_sfd
close(service[i].sfd)
sfd_use = service[i].sfd
TRUE
service[i].protocol == tcp
FALSE
close (connection_sfd)
Chiudo stdin, stdout e e stderr e li sostituisco con *sfd_use*
Sostituisco l'immagine del processo figlio con quella del servizio richiesto
---
execle()
...
TRUE
service[i].mode == wait
FALSE
Salvo in struttura dati il PID del figlio e rimuovo dalla struttura *fd_set* il file descriptor della socket di benvenuto del servizio richiesto
---
FD_CLR(service[i].sfd)
SIGNAL
E' un SIGCHILD
VERO
FALSO
Ottengo il PID del figlio che l'ha scatenato
---
wait()
Ricerco nella struttura dati il servizio di tipo wait di cui il processo figlio si stava occupando
Riaggiungo alla struttura fd_set il file descriptor della socket di benvenuto del servizio trovato e rimuovo il valore PID memorizzato nella struttura
---
FD_SET()