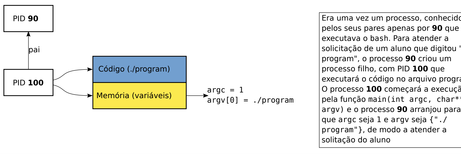
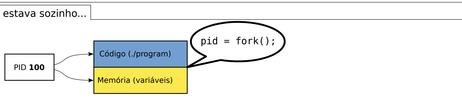
No início...

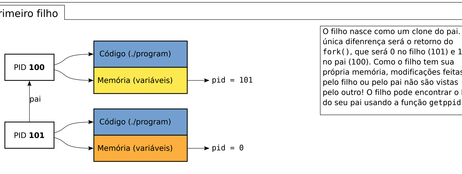


Era uma vez um processo, conhecido pelos seus pares apenas por **90** que executava o bash. Para atender a solicitação de um aluno que digitou "`./program`", o processo **90** criou um processo filho, com PID **100** que executará o código no arquivo program. O processo **100** começará a execução pela função `main(int argc, char** argv)` e o processo **90** arranjou para que argc seja 1 e argv seja {"`./program`"}, de modo a atender a solitação do aluno

100 estava sozinho...

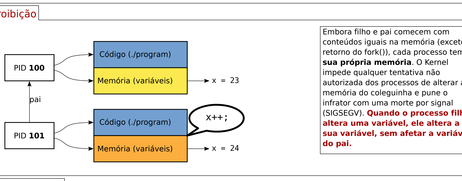


O primeiro filho



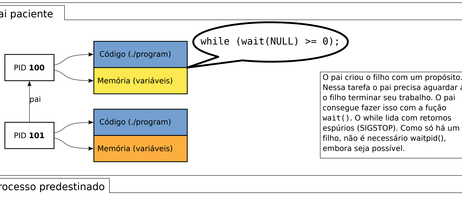
O filho nasce como um clone do pai. A única diferença será o retorno do `fork()`, que será 0 no filho (101) e 101 no pai (100). Como o filho tem sua própria memória, modificações feitas pelo filho ou pelo pai não são vistas pelo outro! O filho pode encontrar o PID do seu pai usando a função `getppid()`.

A proibição



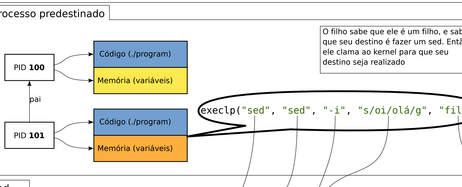
Embora filho e pai comecem com conteúdos iguais na memória (exceto o retorno do fork()), cada processo tem **sua própria memória**. O Kernel impede qualquer tentativa não autorizada dos processos de alterar a memória do coleguinha e pune o infrator com uma morte por signal (SIGSEGV). **Quando o processo filho altera uma variável, ele altera a sua variável, sem afetar a variável do pai.**

O pai paciente



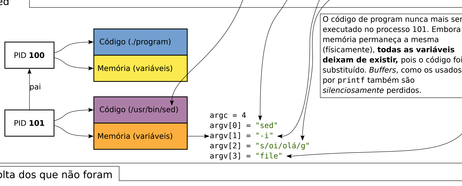
O pai criou o filho com um propósito. Nessa tarefa o pai precisa aguardar até o filho terminar seu trabalho. O pai consegue fazer isso com a fução `wait()`. O while lida com retornos espúrios (SIGSTOP). Como só há um filho, não é necessário waitpid(), embora seja possível.

O processo predestinado



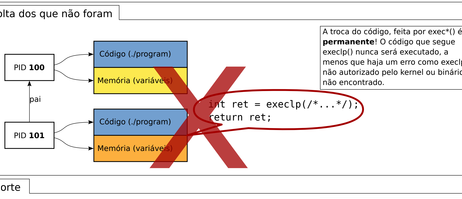
O filho sabe que ele é um filho, e sabe que seu destino é fazer um sed. Então ele clama ao kernel para que seu destino seja realizado

O sed



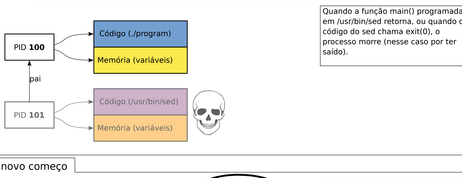
O código de program nunca mais será executado no processo 101. Embora a memória permaneça a mesma (fisicamente), **todas as variáveis deixam de existir,** pois o código foi substituído. *Buffers*, como os usados por `printf` também são *silenciosamente* perdidos.

A volta dos que não foram



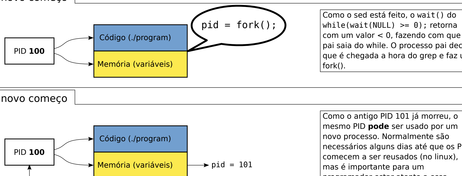
A troca do código, feita por exec*() é **permanente**! O código que segue execlp() nunca será executado, a menos que haja um erro como execlp() não autorizado pelo kernel ou binário não encontrado.

A morte



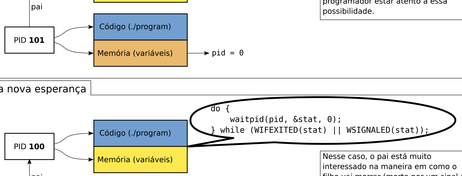
Quando a função main() programada em /usr/bin/sed retorna, ou quando o código do sed chama exit(0), o processo morre (nesse caso por ter saído).

Um novo começo



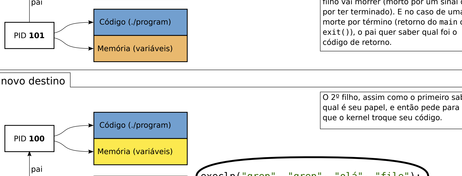
Como o sed está feito, o `wait()` do `while(wait(NULL) >= 0);` retorna com um valor < 0, fazendo com que o pai saia do while. O processo pai decide que é chegada a hora do grep e faz um fork().

Um novo começo



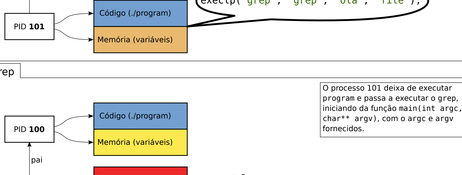
Como o antigo PID 101 já morreu, o mesmo PID **pode** ser usado por um novo processo. Normalmente são necessários alguns dias até que os PIDs comecem a ser reusados (no linux), mas é importante para um programador estar atento a essa possibilidade.

Uma nova esperança



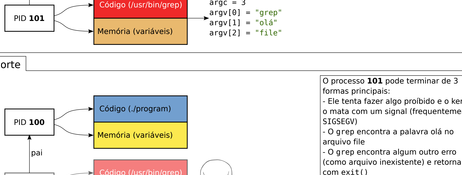
Nesse caso, o pai está muito interessado na maneira em como o filho vai morrer (morto por um sinal ou por ter terminado). E no caso de uma morte por término (retorno do `main` ou `exit()`), o pai quer saber qual foi o código de retorno.

Um novo destino



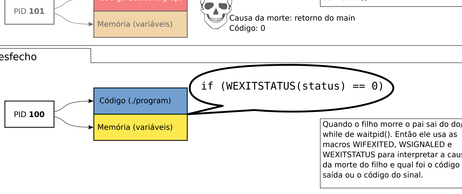
O 2º filho, assim como o primeiro sabe qual é seu papel, e então pede para que o kernel troque seu código.

O grep



O processo 101 deixa de executar program e passa a executar o grep, iniciando da função `main(int argc, char** argv)`, com o argc e argv fornecidos.

A morte



O processo **101** pode terminar de 3 formas principais:
- Ele tenta fazer algo proibido e o kernel o mata com um signal (frequentemente SIGSEGV)
- O grep encontra a palavra olá no arquivo file
- O grep encontra algum outro erro (como arquivo inexistente) e retorna com `exit()`

O desfecho

Quando o filho morre o pai sai do do/while de waitpid(). Então ele usa as macros WIFEXITED, WSIGNALED e WEXITSTATUS para interpretar a causa da morte do filho e qual foi o código de saída ou o código do sinal.