A white text on a white background

Description automatically generated

A computer code with black text

Description automatically generated

A diagram of a memory process

Description automatically generated

A text on a white background

Description automatically generated

Sau khi xài hàm fork(), lúc này cả 2 process con và cha đều chạy song song với fork() return 0 (process con) và return id của process con (process cha). Khi này ta có thể triển khai bằng cách check if (process\_id\_get\_from\_fork == your process num) để xử lý

Thoát 1 process bằng hàm #include <signal.h> + int kill(childPid,SIGKILL);

Hoặc kill bằng command : kill -9 process\_id

**Process monitor**

**A white background with black text

Description automatically generated**

A close up of a text

Description automatically generated

Pid\_t wait(int \*status); hàm wait ở đây sẽ trả về 1 giá trị dựa vào exit status

A computer screen with blue text

Description automatically generated

* Nếu exit(exitStatus) thành công, thì exit status sẽ ghi vào bit thứ 8-15

VD exit(3) -> no error -> 00000011 | 00000000 = 768

A close-up of a credit card

Description automatically generated

* Nếu bị kill –(num) thì 7 bit từ 0 -6 sẽ chứa giá trị num, còn bit thứ **7** sẽ bật thành 1 tượng trưng cho flag

VD bị kill bởi : kill -8 process\_id -> mã lỗi 136 tương ứng với 00000000 | **1** 0001000

Nếu có nhiều hơn 1 child process, thì khi wait hàm sẽ đợi thông tin từ process nào xong gần nhất, nếu muốn đợi cụ thể process nào thì dùng waitpid(pid, &status, flag like : WNOHANG (this is non-blocking-call to waitpid))

Sử dụng WNOHANG sẽ làm cho process cha không chờ để nhận giá trị exit của process con (ko có thì bỏ qua luôn)