Referat

„Kiedy proces sieroty przyjmuje pid inne niż init?”

Wiktoria Koziej

III rok informatyka st.

# Czym jest proces sieroty?

Prawie każdy proces w systemie UNIX jest tworzony przez inny proces, który staje się jego przodkiem. Przodek może zakończyć swoje działanie przed zakończeniem swojego potomka. Taki proces potomny, którego przodek już się zakończył, nazywany jest *sierotą* (ang. orphan).

Procesy-sieroty są przekazywane do specjalnego procesu o nazwie "init" lub "systemd" w zależności od używanego systemu UNIX. Ten proces ma identyfikator PID równy 1 i jest odpowiedzialny za zarządzanie procesami systemowymi. Kiedy proces-sierota zostanie przekazany do "init" lub "systemd", zyskuje on nowego rodzica i nie jest już związany z pierwotnym terminalem, który utracił.

Ten mechanizm zapewnia, że ​​procesy-sieroty nie są pozostawiane bez nadzoru, co umożliwia systemowi operacyjnemu UNIX utrzymanie porządku i kontroli nad procesami nawet po zakończeniu sesji użytkownika.

Obraz zawierający tekst, zrzut ekranu, Czcionka, pismo odręczne

Opis wygenerowany automatycznie

Rysunek 1 Przykładowy proces tworzenia procesu-sieroty

# Funkcja Prctl()

Funkcja prctl() służy do manipulowania różnymi aspektami zachowania bieżącego wątku lub procesu.

Należy zauważyć, że nieuważne korzystanie z niektórych operacji prctl() może wprowadzić zamieszanie w środowisku uruchomieniowym przestrzeni użytkownika, dlatego te operacje powinny być używane ostrożnie.

Aby wykorzystać funkcję prctl() należy dodać odpowiedni plik nagłówkowy tj. <linux/prctl.h> .

W tym pliku nagłówkowym znajduje się interesująca nas operacja PR\_SET\_CHILD\_SUBREAPER istniejąca od Linuxa 3.4.

Proces subreaper pełni rolę init(1) dla swoich procesów potomnych. Gdy proces staje się sierotą (tj. jego bezpośredni rodzic kończy działanie), ten proces zostanie przypisany do najbliższego żyjącego przodka będącego subreaperem. Następnie wywołania getppid(2) w procesie sierocym zwrócą PID procesu subreapera, a gdy sierota zakończy działanie, to proces subreapera otrzyma sygnał SIGCHLD i będzie mógł użyć wait(2), aby sprawdzić status zakończenia procesu sieroty.

Ustawienie atrybutu "child subreaper" nie jest dziedziczone przez procesy potomne utworzone przez fork(2) i clone(2).

Wykorzystując tę operację możemy napisać program odzwierciedlający uzyskanie przez proces-sierotę pid innego niż init.

#include <stdio.h>

#include <sys/types.h>

#include <sys/prctl.h>

#include <unistd.h>

#include <sys/wait.h>

#include <stdlib.h>

int main(void)

{

int \*status;

int i=0;

prctl(PR\_SET\_CHILD\_SUBREAPER, 1, 0, 0, 0);

perror("PARENT:Set");

printf("PARENT: %d : my dad : %d\n", getpid(), getppid());

if(fork() != 0)

{

while(1)

{

wait(status);

if(++i == 2)

{

break;

}

}

int p = 1;

prctl(PR\_GET\_CHILD\_SUBREAPER, &p);

printf("PARENT : %d\n",p);

printf("PARENT Exiting\n");

}

else

{

printf("Before CHILD: %d: my dad %d\n",getpid(), getppid());

if(fork() == 0)

{

int p = 1;

printf("Before grandchild: %d: my dad %d\n",getpid(), getppid());

sleep(2);

printf("After grandchild: %d: my dad %d\n",getpid(), getppid());

prctl(PR\_GET\_CHILD\_SUBREAPER, &p);

printf("After grandchild : %d\n",p);

printf("Grandchild Exiting\n");

exit(0);

}

else

{

int p = 1;

prctl(PR\_GET\_CHILD\_SUBREAPER, &p);

printf("After CHILD : %d\n",p);

printf("After CHILD: %d: my dad %d\n",getpid(), getppid());

printf("CHILD Exiting\n");

exit(1);

}

}

return 0;

}

Proces PARENT staje się subreaperem, ustawiając PR\_SET\_CHILD\_SUBREAPER. Proces PARENT rozgałęzia się i tworzy proces CHILD. Proces CHILD rozgałęzia się i tworzy proces Grandchild. Proces CHILD próbuje użyć PR\_GET\_CHILD\_SUBREAPER i otrzymuje 0. Ponieważ prctl() ma tylko jedną instancję, nie zostanie zachowane w procesach utworzonych przez fork(). Proces CHILD kończy działanie, sprawiając, że proces Grandchild staje się procesem sierotą. Ponieważ proces PARENT został ustawiony jako SUBREAPER, proces Grandchild staje się dzieckiem procesu PARENT, a status zakończenia procesu Grandchild jest odbierany przez proces PARENT, ponieważ proces Child zakończył działanie.

# Bibliografia

* 1. <https://man7.org/linux/man-pages/man2/prctl.2.html>
  2. <https://wazniak.mimuw.edu.pl/index.php?title=SOP_lab_nr_7>
  3. <https://stackoverflow.com/questions/56856275/is-there-some-short-example-of-how-to-use-prctl-when-it-comes-to-setting-subre>