Maryna Lukachyk (308294)

Laboratorium 4 – Procesy

I. Implementacja własnej powłoki

Funkcje systemowe:

- **waitpid(2)**

#include <sys/types.h>

#include <sys/wait.h>

Funkcja waitpid zawiesza wykonywanie bieżącego procesu dopóki potomek określony przez pid nie zakończy działania lub dopóki nie zostanie dostarczony sygnał, którego akcją jest zakończenie procesu lub wywołanie funkcji obsługującej sygnały.

Wartość pid:

< -1 co oznacza oczekiwanie na dowolny proces potomny, którego ID grupy procesów jest równy wartości bezwzględnej pid.

-1 co oznacza oczekiwanie na dowolny proces potomny; jest to takie samo zachowanie, jakie wykazuje wait.

0 co oznacza oczekiwanie na każdy proces potomny, którego ID grupy procesu jest równe ID grupy procesu wywołującego funkcję.

> 0 oznacza oczekiwanie na potomka, którego ID procesu jest równy wartości pid.

BŁĘDY :

ECHILD jeśli proces o zadanym pid nie istnieje lub nie jest potomkiem procesu wywołującego.

EINVAL jeśli argument options jest niepoprawny.

EINTR jeśli WNOHANG nie było ustawione, a został przechwycony niez-ablokowany sygnał lub SIGCHLD.

- **fork(2)**

#include <sys/types.h>

#include <unistd.h>

fork tworzy proces potomny, który różni się od procesu macierzystego jedynie swoimi numerami PID i PPID oraz tym, że w rzeczywistości użycie przez niego zasobów jest ustawione na 0. Blokady plików i oczekujące sygnały nie są dziedziczone.

BŁĘDY :

EAGAIN fork nie mógł zaalokować ilości pamięci wystarczającej do skopiowania tablic stron rodzica i dla struktury zadania dla potomka. **ENOMEM** fork nie potrafił zaalokować niezbędnych struktur jądra z powodu niedostatecznej ilości pamięci.

Funkcje operujące na stringach:

- **strcmp()**

#include <string.h>

int strcmp(const char *s1, const char *s2);

Funkcja strcmp poruwnuje dwa stringi str1 i str2;

Zwraca int >,< lub = 0;

- **Strtok()**

#include <string.h>

char *strtok(char *s, const char *delim);

`Słowo' jest to niepusty ciąg znaków, które nie występują w łańcuchu delim, poprzedzony znakiem \0 lub znakiem występującym w delim.

Funkcja strtok() służy do rozkładania ciągu znaków s na słowa. Pier- wsze odwołanie do strtok() powinno posiadać s jako pierwszy argument. Następne wywołania powinny mieć jako pierwszy argument NULL. Każde wywołanie zwraca wskaźnik do następnego słowa lub NULL, gdy nie ma już więcej słów.

- **Strchr()**

#include <string.h>

char *strchr(const char *s, int c);

Funkcja strchr() zwraca wskaźnik do pierwszego wystąpienia znaku c w łańcuchu znaków s.

"Znak" oznacza tutaj "bajt" - funkcje te nie działają ze znakami wielobajtowymi.

Funkcje zarządzające pamięcią:

#include <stdlib.h>

- **Realloc()**

  void *realloc(void *ptr, size_t size);

  zmienia rozmiar bloku pamięci wskazywanego przez ptr na size bajtów. Zawartość nie zostanie zmieniona w zakresie poniżej minimum ze starego i nowego rozmiaru; nowo przydzielona pamięć nie zostanie zainicjalizowana.

- **Free()**

  void free(void *ptr);

  zwalnia obszar pamięci wskazywany przez ptr, który został wcześniej przydzielony za pomocą wywołania malloc(), calloc() lub realloc(). W przeciwnym przypadku, lub gdy free(ptr) zostało już wcześniej wywołane, funkcja zachowa się w sposób nieokreślony. Jeśli ptr jest równe NULL, nie zostanie wykonana żadna operacja.

Inne funkcje:

Exec() ~ execvp()

exec [command [argument ...]]

The exec utility shall open, close, and/or copy file descriptors as specified by any redirections as part of the command.

execvp - uruchomienie pliku

#include <unistd.h>

extern char **environ;

int execvp(const char *file, char *const argv[]);

Funkcje execv oraz execvp udostępniają tablicę wskaźników do zakończonych znakami NUL łańcuchów, które reprezentują listę argumentów dostępnych dla wykonywanego programu. Pierwszy argument, zgodnie z konwencją, powinien wskazywać na nazwę pliku powiązaną z wykonywanym plikiem. Tablica wskaźników musi być zakończona wskaźnikiem NULL.

II. Ustawiania limitów procesu

Funkcje systemowe:

- **Setrlimit()** - pobranie/ustawienie limitów i zużycia

```
#include <sys/time.h>
#include <sys/resource.h>
#include <unistd.h>
    int setrlimit(int resource, const struct rlimit *rlim);
```

getrlimit i setrlimit odpowiednio pobierają i ustawiają limity zasobów.

BŁĘDY :

EFAULT rlim lub usage wskazuje poza dostępną przestrzeń adresową.

EINVAL getrlimit lub setrlimit zostało wywołane ze złym resource, lub też getrusage zostało wywołane ze złym who.

EPERM Użytkownik nie będący superużytkownikiem próbuje używać setr- limit() do zwiększenia miękich lub twardych limitów, lub też superużytkownik próbuje zwiększyć RLIMIT_NOFILE powyżej maksimum jądra.

- **_Exit** - zakończenie bieżącego procesu

Funkcja _exit "natychmiast" kończy proces, z którego została wywołana. Wszystkie przynależące do procesu otwarte deskryptory plików są zamykane;

▪ Argumenty programów w C, zmienne argc i argv

Pierwszy parametr, o zwyczajowej nazwie ***argc***, jest liczbą argumentów podanych podczas wywołania programu. Pierwszym z tych argumentów, o numerze 0, jest zawsze nazwa wywoływanego programu. Tak więc ***argc*** jest zawsze co najmniej jeden. Zmienna ***argv*** wygląda trochę tajemniczo, ale tak naprawdę jest tablicą napisów zawierających kolejne argumenty wywołania: ***argv[0]***, ***argv[1]***, ..., ***argv[argc-1]***; indeksowanie rozpoczyna się od zera, dlatego ostatnią wartścią indeksu jest ***argc-1*** a *nie* ***argc***. Tablica ta jest podobna do tablicy, nazywanej zwykle ***args***, w Javie. Ponieważ, jak się przekonamy, tablice w C/C++ nie są obiektami i nie zawierają odpowiednika zmiennej obiektowej ***length*** z Javy, więc trzeba było przekazać tablicę napisów reprezentujących argumenty oraz, osobno, rozmiar tej tablicy (poprzez parametr ***argc***).

▪ Funkcje konwertujące: atoi(3)

przekształcenie łańcucha na wartość całkowitą

#include <stdlib.h>

int atoi(const char *nptr);

Funkcja atoi() przekształca początkową część łańcucha wskazywanego przez nptr na int. Działa tak samo, jak strtol(nptr, (char **)NULL, 10); z wyjątkiem tego, że atoi() nie wykrywa błędów.

▪ Struktury: struct rlimit

```
struct rlimit {
    rlim_t rlim_cur;   /* ograniczenie miękkie */
    rlim_t rlim_max;   /* ograniczenie sztywne (górna
                          granica dla rlim_cur) */
};
```