Dokumentacja programu Mini-cron

Marcin Górecki
Natalia Orłowska

1. Ogólny opis działania programu

Program Mini-cron jest uproszczoną wersją uniksowego demona Cron służącego do harmonogramowaniu i wykonywaniu zaplanowanych zadań. Mini-cron wczytuje z pliku czas oraz polecenie do wykonania z opcją wypisania nazwy wykonywanego polecenia na jeden ze strumieni: standardowe wyjście, wyjście błędów lub na oba jednocześnie, sortuje je chronologicznie, a następnie zasypia i budzi się, kiedy nadejdzie czas na wykonanie kolejnego zadania. Demon zapisuje do logu systemowego informacje o uruchomieniu zadania i kodzie wyjścia, a także listę pozostałych do wykonania zadań (wyłącznie po otrzymaniu sygnały SIGUSR2). Obsługiwane są potoki. Program został napisany w języku C z zastosowaniem API Linuksa.

2. Sposób wywołania

Mini-cron można uruchomić przez wpisanie w terminalu polecenia

```
./minicron <taskfile> <outfile>
```

(pod warunkiem, że znajdujemy się w katalogu programu).

`taskfile` to plik przechowujący zadania do uruchomienia przez demon w następującym formacie

```
<hour>:<minute>:<command>:<info>
```

Przy czym `command` jest dowolnym programem lub potokiem programów, a parametr `info` przyjmuje następujące wartości:

- 0 – treść wypisywana przez polecenie jest przekazywana na standardowe wyjście (stdout)
- 1 – treść wypisywana przez polecenie jest przekazywana na wyjście błędów (stderr)
- 2 – treść wypisywana przez polecenie jest przekazywana zarówno na standardowe wyjście, jak i na wyjście błędów

`outfile` jest plikiem zawierającym informacje zlecane przez użytkownika wraz z informacją o poleceniu, które je wygenerowało.

3. Opis bibliotek i funkcji programu

Biblioteka **command_list.h**

Zawiera implementację struktury `SingleCommand`, która przechowuje zadania do wykonania oraz wskaźnik `CommandList` na listę zadań. Struktura `SingleCommand` zawiera czas wykonania zadania (zmienna typu `time_t` z biblioteki **time.h** języka C) `commandTime`, wskaźnik na tablicę znaków `command` przechowujący polecenie, parametr `info` (opisany w pkt. 2 dokumentacji), oraz wskaźnik na następne polecenie. Dodatkowo znajdują się tam funkcje obsługujące listę zadań:

```
void addCommand(CommandList* root, int hour,
                                   int minute,
                                   char* command,
                                   int info,
                                   time_t now)
```

Funkcja dodaje zadanie do listy poleceń. Jeśli zaplanowany czas wykonania zadania jest mniejszy niż czas dodawania zadania (`now`), funkcja planuje zadanie na kolejny dzień.

```
void createCommandList(CommandList* root, int taskfile_fd)
```

Funkcja odczytuje linia po linii zawartość pliku przypisanego deskryptorowi `taskfile_fd`, zapisuje każdą z linii do bufora i tworzy z niej polecenie, które dodaje do listy. Na koniec lista jest sortowana przy użyciu funkcji `sort`. Wszelkie błędy zapisywane są do logu systemowego.

```
void deleteRoot(CommandList* root)
```

Funkcja usuwa korzeń listy i przypisuje wskaźnikowi listy następny element.

```
SingleCommand getNext(CommandList root)
```

Funkcja zwraca kolejne (tzn. najbliższe) zadanie do wykonania z listy.

```
void clearList(CommandList* root)
```

Funkcja usuwająca wszystkie elementy listy.

```
CommandList last(CommandList root)
```

Funkcja zwraca ostatni element listy zadań.

```
void sort(CommandList root)
```

Funkcja sortująca elementy listy chronologicznie (tzn. w kolejności od najbliższego czasu wykonania).

```
void swap(CommandList a, CommandList b)
```

Funkcja zamieniająca kolejność dwóch elementów w liście.

Biblioteka **daemon_functions.h**

Biblioteka zawiera funkcje, z których korzysta demon.

```
int getNextSeconds(SingleCommand cmd)
```

Funkcja zwraca różnice sekund między aktualnym czasem a czasem wykonania najbliższego polecenia.

```
void runCommand(char* cmd, int info, int out_fd)
```

Funkcja tworzy proces potomny, wykonujący wskazywane przez `cmd` polecenie, zapisując zwróconą treść do pliku, w pliku wyjściowym identyfikowanym przez deskryptor `out_fd`.

```
time_t setNow(void)
```

Funkcja zwraca wartość typu `time_t` reprezentującą aktualny czas.

```
void saveToSyslog(CommandList root)
```

Funkcja zapisuje zadania oczekujące na wykonanie do logu systemowego.

Program główny **main.c**

Program główny otwiera pliki podane jako parametry przez użytkownika oraz log systemowy. Następnie wywołuje funkcję tworzącą listę poleceń. Demon zasypia i budzi się, gdy nadchodzi czas wykonania kolejnego zadania, a następnie usuwa to polecenie z listy. Wykonanie programu trwa dopóki dostępne są zadania w liście lub do otrzymania sygnału SIGINT. Zamykane są pliki i log systemowy.

- Obsługa sygnałów

  Każde wzbudzenie sygnału SIGUSR1 oraz SIGUSR2 wywołuje funkcję ich obsługi `handler`, która definiuje odpowiednie zachowania. Otrzymanie sygnału SIGUSR1 prowadzi do aktualizacji listy poleceń, zaś SIGUSR2 powoduje zapisanie zadań pozostałych do wykonania w logu systemowym.