Projekt: zarządzanie interaktywnymi kopiami zapasowymi

Jak mówi stare informatyczne przysłowie „ludzie dzielą się na tych, którzy robią kopie zapasowe oraz tych, którzy będą je robić", jednak ręczne tworzenie takich kopii bywa czasochłonne i mało ciekawe. Dlatego stworzymy system, który automatycznie zarządza tworzeniem backupów na komputerze.

Po uruchomieniu program wypisuje listę dostępnych komend i czeka na polecenie od użytkownika. Wprowadzenie błędnej komendy skutkuje odpowiednim komunikatem, ale nie kończy działania programu.

Utworzenie kopii

Polecenie `add <source path> <target path>` uruchamia tworzenie kopii folderu `<source path>` w miejscu wskazanym przez `<target path>`. Można podać wiele ścieżek docelowych, oddzielając je spacją - wtedy kopie zostaną utworzone w kilku lokalizacjach.

Jeżeli któryś z podanych katalogów docelowych nie istnieje, program powinien go utworzyć. Jeśli natomiast katalog docelowy już istnieje, przed rozpoczęciem kopiowania musi być pusty. W przeciwnym razie program powinien zgłosić błąd.

Tworzenie kopii polega na przekopiowaniu całej struktury katalogów i plików z folderu źródłowego do wszystkich wskazanych katalogów docelowych. Program powinien obsługiwać zarówno zwykłe pliki, jak i dowiązania symboliczne.

Jeżeli dowiązanie symboliczne w katalogu źródłowym zawiera ścieżkę bezwzględną prowadzącą do tego katalogu, to w kopii należy utworzyć symlink wskazujący na odpowiedni plik w katalogu docelowym. We wszystkich pozostałych przypadkach ścieżka w dowiązaniu powinna zostać skopiowana bez zmian.

Po zakończeniu tworzenia kopii program rozpoczyna monitorowanie katalogu źródłowego. Wszystkie zmiany - obejmujące pliki, dowiązania symboliczne oraz podfoldery - są na bieżąco odzwierciedlane w katalogach docelowych.

Monitorowanie zmian kończy się, gdy użytkownik zakończy wybraną kopię, zamknie program lub gdy katalog źródłowy zostanie usunięty.

Kluczowe jest zachowanie wysokiej responsywności programu - użytkownik może wprowadzać kolejne polecenia tuż po zaakceptowaniu poprzedniego, niezależnie od trwających operacji kopiowania. Każdy katalog docelowy obsługuje niezależny podproces, zapewniający równoległe wykonywanie zadań i brak przestojów w obsłudze poleceń.

Program powinien uniemożliwić utworzenia:

- kopii katalogu wewnątrz jego samego (w przeciwnym wypadku skończyło by się to niekończącą się kaskadą tworzenia plików),
- kilku kopii o tych samych ścieżkach źródłowej i docelowej.

Zakończenie tworzenia kopii

Polecenie `end <source path> <target paths>` przerywa tworzenie kopii zapasowych w podanych katalogach docelowych. Zawartość tych folderów pozostaje nienaruszona.

Listowanie aktywnych kopii

Polecenie `list` wypisuje na standardowe wyjście podsumowanie, pokazujące które foldery mają kopie zapasowe i dokąd są one zapisywane.

Przywracanie kopii

Polecenie `restore <source path> <target path>` przywraca wybraną kopię zapasową. Program kopiuje całą strukturę katalogów i pliki z folderu docelowego do źródłowego, a także usuwa z folderu źródłowego te pliki, których nie ma w kopii. Aby przyspieszyć operację, kopiowane są tylko pliki, które zmieniły się od czasu utworzenia kopii. Polecenie przyjmuje tylko jedną ścieżkę docelową.

Należy pamiętać o ewentualnej zmianie ścieżki przy kopiowaniu linków symbolicznych.

Komenda powinna działać w sposób blokujący - wydanie kolejnych poleceń jest możliwe dopiero po zakończenia przywracania kopii.

Kończenie działania programu

Komenda `exit` kończy pracę programu. Program musi zwolnić wykorzystywane zasoby oraz zakończyć pracę wszystkich procesów potomnych. Program powinien również poprawnie zakończyć pracę w przypadku trzymania sygnały `SIGINT` oraz `SIGTERM`. Reszta sygnałów powinna być ignorowana.

Uwagi i wskazówki

- Nie wolno korzystać z funkcji `system`.
- Nazwy folderów mogą zawierać spacje, z tego powodu wymagana jest obsługa cudzysłowów analogiczna do tej znanej z bash'a.
- Do śledzenia zmian w folderze źródłowym można wykorzystać API `inotify`. Więcej można przeczytać w następujących stronach manuala:
  - `man 7 inotify`
  - `man 2 inotify_init`
  - `man 2 inotify_add_watch`
  - `man 2 inotify_rm_watch`
- Do porównywania ścieżek przydaje się funkcja `realpath`.

Punktacja

Projekt będzie oceniany w pierwszej kolejności na podstawie w pełni działających funkcjonalności zgodnie z tabelą poniżej. Liczba punktów może zostać obniżona ze względu na:

- Błędy związane z niezwalnianiem zasobów, nieobsługiwaniem błędów funkcji systemowych i inne, mogące prowadzić do nieprawidłowego działania programu
- Styl kodu - np. zbędne zmienne globalne, bardzo długie funkcje, tzw. „magic numbers" itd.
- Nieumiejętność wyjaśnienia przez studenta działania programu

Należy mieć na uwadze zwłaszcza ostatni punkt - praca musi pozostać samodzielna. W przypadku gdy student nie będzie potrafił odpowiedzieć na pytania prowadzącego dotyczące jego kodu, punkty mogą zostać wyzerowane.

1. [1 p.] Możliwa jest kopia zapasowa do pojedynczego folderu docelowego, bez dalszego monitorowania i synchronizacji zmian.
2. [4 p.] Oczekiwanie na zmiany i odzwierciedlanie ich.
3. [3 p.] Możliwość dodania wielu katalogów docelowych.
4. [1 p.] Kończenie tworzenia kopii.
5. [1 p.] Listowanie aktywnych kopii.
6. [2 p.] Przywracanie kopii (w przypadku kopiowania wszystkich plików można otrzymać maksymalnie jeden punkt).

Komenda `exit` jest wymagana w każdym programie.