Zadanie polega na dodaniu dwóch nowych wywołań systemowych:

- `PM_CHANGE_PARENT` z funkcją biblioteczną `int changeparent()` oraz
- `PM_GETOPPID` z funkcją biblioteczną `pid_t getoppid(pid_t pid)`.

Funkcje biblioteczne powinny być zadeklarowane w pliku `unistd.h`.

Zmiana rodzica

Funkcja `PM_CHANGE_PARENT` zmienia rodzica procesu, który wywołał funkcję, na jego „dziadka", czyli rodzica dotychczasowego rodzica, pod warunkiem, że

- dotychczasowy rodzic nie jest procesem `init` oraz
- rodzic nie oczekuje na zakończenie potomka, czyli nie wykonuje funkcji `wait()`.

Jeśli te warunki nie są spełnione, to rodzic pozostaje bez zmian.

Funkcja `changeparent()` przekazuje w wyniku 0, jeśli rodzic został zmieniony, a −1 w przypadku niepowodzenia. Jeśli rodzicem procesu jest `init`, to funkcja ustawia `errno`, na `EACCES`, a jeśli rodzic oczekuje na zakończenie potomka, to na `EPERM`.

Oryginalny rodzic

Funkcja `PM_GETOPPID` przekazuje w wyniku identyfikator oryginalnego rodzica dla zadanego procesu.

Jeśli proces nie zmieniał rodzica, to oryginalny rodzic jest aktualnym rodzicem. Nawet jeśli proces zmieniał rodzica wiele razy, to oryginalny rodzic pozostaje niezmieniony.

Funkcja `getoppid(pid_t pid)` przekazuje w wyniku identyfikator oryginalnego rodzica procesu o identyfikatorze `pid`, o ile rodzic jeszcze działa, a w przeciwnym przypadku przekazuje 0.

Jeśli proces o podanym `pid` nie istnieje, funkcja przekazuje w wyniku −1 i ustawia `errno` na `EINVAL`.

Format rozwiązania

Poniżej przyjmujemy, że `ab123456` oznacza identyfikator studenta rozwiązującego zadanie. Należy przygotować łatkę (ang. *patch*) ze zmianami w katalogu `/usr`. Plik zawierający łatkę o nazwie `ab123456.patch` uzyskujemy za pomocą polecenia

```
diff -rupN oryginalne-źródła/usr/ moje-rozwiązanie/usr/ > ab123456.patch
```

gdzie `oryginalne-źródła` to ścieżka do niezmienionych źródeł MINIX-a, natomiast `moje-rozwiązanie` to ścieżka do źródeł MINIX-a zawierających rozwiązanie. Tak użyte polecenie `diff` rekurencyjnie przeskanuje pliki ze ścieżki `oryginalne-źródła/usr`, porówna je z plikami ze ścieżki `moje-rozwiązanie/usr` i wygeneruje plik `ab123456.patch`, który podsumowuje różnice. Tego pliku będziemy używać, aby automatycznie nanieść zmiany na czystą kopię MINIX-a, gdzie będą przeprowadzane testy rozwiązania. Więcej o poleceniu `diff` można dowiedzieć się z podręcznika (`man diff`).

Umieszczenie łatki w katalogu `/` na czystej kopii MINIX-a i wykonanie polecenia `patch -p1 < ab123456.patch` powinno skutkować naniesieniem wszystkich oczekiwanych zmian wymaganych przez rozwiązanie. Należy zadbać, aby łatka zawierała tylko niezbędne różnice.

Po naniesieniu łatki zostaną wykonane polecenia:

- `make && make install` w katalogach `/usr/src/minix/servers/pm` oraz `/usr/src/lib/libc`,
- `make do-hdboot` w katalogu `/usr/src/releasetools`,

- `reboot`.

Rozwiązanie w postaci łatki `ab123456.patch` należy umieścić na Moodlu.

Uwagi

- Serwer PM przechowuje informacje o procesach w tablicy `mproc` zadeklarowanej w pliku `mproc.h`. Działające procesy mają ustawioną flagę `IN_USE`.
- Warto przeanalizować jak PM realizuje wywołania systemowe. Więcej informacji o działaniu tego serwera będzie na laboratorium 7.
- Należy samodzielnie przetestować rozwiązanie. Jeden z podstawowych scenariuszy jest następujący: uruchamiamy proces A, który uruchamia proces B, po czym wykonuje `wait()`. Proces B uruchamia proces C, po czym wykonuje `sleep()`. Proces C zmienia rodzica (`changeparent()`). Upewniamy się, że nowym rodzicem C jest A (`getppid()`) oraz że oryginalny rodzic został poprawnie zapamiętany, czyli `getoppid(pid_C)` zwraca `pid` B.
- Nie przyznajemy punktów za rozwiązanie, w którym łatka nie nakłada się poprawnie, które nie kompiluje się lub które powoduje *kernel panic* podczas uruchamiania systemu.