Raport, laboratorium 3

Tomasz Urban

25 października 2023

1 Zadanie 3

1.1 Utworzone skrypty

- pilot.sh - skrypt symulujący działanie pilota,
- projektor.sh - skrypt symulujący działanie projektora

1.2 Opis działania

1.2.1 pilot.sh

Argumenty wywołania Numer PID procesu, do którego ma zostać wysłany sygnał
Przebieg Skrypt sprawdza, czy podano prawidłową liczbę argumentów wejściowych. Jeżeli nie, kończy działanie. W przeciwnym przypadku wchodzi do pętli. Na początku pętli użytkownik pytany jest o sygnał jaki chce wysłać do procesu o podanym numerze PID. W zależności od wybranej przez użytkownika opcji poszukiwany jest numer odpowiadający danemu sygnałowi. (z użyciem poelenia kill i opcji -l). Polecenie kill jest ponownie używane do wysłania sygnału do określonego procesu.

1.2.2 projektor.sh

Argumenty wywołania Brak argumentów wywołania.
Przebieg Do przechwytywania sygnałów wykorzystano funkcję trap. Dla każdego rodzaju sygnału przeznaczono inną linijkę skryptu z funkcją trap. Reakcje skrypty na poszczególne zewnętrzne sygnały:

- SIGFPE - proces ignoruje ten sygnał,
- SIGUSR1 - skrypt wywołuje funkcję signal_controller, która w zależności od wartości zmiennej signal_count kończy działanie, bądź zaczyna odliczanie na następny sygnał SIGUSR1 (czas minie signal_count jest zerowany, jeśli proces otrzyma sygnał przed upłynięciem czasu, kończy swoje działanie)

Reakcje skryptu na różne sygnały

- SIGKILL- proces jest zabity,
- SIGSTOP- proces jest zastopowany (sprawdzone z użyciem polecenia PS i stanu procesu),
- reakcja procesu na kombinację klawiszy Ctr+Z jest taka sama jak na sygnał SIGCONT,
- SIGCONT- proces jest kontunuowany, działanie jest bardziej podobne do polecenia fg

2 Zadanie 4

2.1 Działanie

Skrypt node1.sh co sekundę wypisuje na wyjście tekst "Przykladowy strumien" Trójelementowy potok uruchomiono za pomocą następującego polecenia:

```
./node1.sh | cat | cat
```

Wykorzystując polecenie PS sprawdzono relacje pomiędzy procesami wchodzącymi w skład potoku. Wnioski:

- procesy znajdują się w tej samej grupie procesów (GID),
- procesy mają tego samego rodzica (PPID)

3 Zadanie 5

W zadaniu wykorzystano polecenia nice, renice i top. Polecenie top pozwoliło na monitoring użycia czasu procesora.

Do wykonania zadania użyto:

- programu procesObliczeniowy, który w nieskokńczonej bętli wykonuje operację mnożenia,
- skryptu obliczeniaSkrypt.sh uruchamiającego w tle program podany jako argument wywołania liczbę razy również określoną przez argument wywołania. Proces wywoływany jest przy użyciu polecenia nice. Priorytet określano przy użyciu numeu kolejnej iteracji.

Działanie przetestowano na komputerze diablo. Po uruchumieniu skryptu a w konsekwencji odpowiedniej liczby procesów obliczeniowych użyto polecnia top z flagą -Uturban, w celu obejrzenia czasu użycia procesora przez poszczególne procesy.

Procesy o niższej wartości liczby określającej priorytet (czyli o większym priorytecie) przez większą ilość czasu korzystały z procesora.

W nsatępnym etapie, z użyciem polecenia renice zmieniano priorytety procesu. Po obniżeniu procesu z użyciem polecenia:

```
renice -n <nowy_priorytet> -p <identyfikator_procesu>
```

Zaczął otrzymywać mniejszy przydział czasu użycia procesora.

Z drugiej strony zwiększenie priorytety nie było możliwe.

4 Zadanie 6

Po ustawieniu limitu dostępnych procesów na 2, niemożliwe było uruchomienie w terminalu jakiegokolwiek procesu. Rozwiązanie problemu polegało na wyłączeniu terminala. Po ponownym właczeniu limit powracał do początkowej wartości. Przy limicie niemożliwe było nawet uruchamianie programów jako admistrator (z użyciem sudo). Przy ustawionym limicie, przy próbie uruchomienia programu pojawiał się komunikat:

```
bash: fork: retry: Resource temporarily unavailable
```

Użytkownikowi nie przydzielono wystarczających zasobów do uruchomienia nowego procesu.

Z racji tego, że na moim komputerze było uruchomioych wiele procesów w tle zdecydowałem się na utworzenie nowego użytkownika. Dalsze etapy tego zadania były wykonywane jako nowy użytkownik.

Po ponownym uruchomieniu terminala komend, zmianie użytkownika, utworzono skrypt wywołujący samego siebie w tle. W kolejnym kroku ustawiono limit procesów dla użytkownika na poziomie 8. Dla niższych wartości ponownie pojawiał się komunikat, że ilość zasobów jest niewystarczająca (podobnie jak dla limitu 2). Mogło być to związane z ilością procesów koniecznych do uruchomienia skryptu (kolejno fork, uruchomienie skryptu i wyświetlenie komunikatu z użyciem polecenia echo).

Po uruchomieniu wspomnianego skryptu, dla maksymalnej ilości procesów równej 8, program uruchamiał się wielokrotnie. Niemożliwe było zatrzymanie zapętlenia z użyciem sygnału SIGKILL. Sytuację rozwiązało dopiero wyłączenie terminala.

5 Zadanie 7

Stworzono program wykonujący operację ikrementacji zmiennej i w nieskończonej pętli. Z użyciem programu top sprawdzono czas procesora przydzielanemu procesowi. Wywołany proces przez większość czasu otrzymuje prawie 100% czasu procesora - znajduje się na szczycie listy top.

5.1 Modyfikacja

W następnym etapie dodano w pętli polecenie sleep z czasem opóźnienia równym 1. Ponownie monitorowano procesy poleceniem top. Częstotliwość odświeżania ustawiono za pomocą flagi -d. Czas odświeżania ustawiono na 0.5 sekundy. Proces obliczeniowy po modyfikacji otrzymywał niewielką ilość czasu. Nie znajdował się na szczycie. Inne procesy działające w tle znajdowały się wyżej w hierarchii.

6 Zadanie 8

Zrealizowano program reagujący na następujące sygnały wejściowe:

- SIGALRM,
- SIGUSR1,
- SIGUSR2,
- SIGTERM

Reakcje na sygnały:

- SIGALRM - ten sygnał jest ignorowany przez proces,
- SIGTERM - sygnał powoduje zakończenie działania procesu,
- SIGUSR1 - sygnał powoduje, że ptogram wraca do normalnej pracy,
- SIGUSR2 - sygnał blokuje reakcję procesu na sygnały

Do wywołania poszczególnych sygnałów użyto następujących komend:

- kill -$(kill -l SIGALRM) numer_procesu
- kill -$(kill -l SIGTERM) numer_procesu
- kill -$(kill -l SIGUSR1) numer_procesu
- kill -$(kill -l SIGUSR2) numer_procesu

Reakcja na sygnał SIGUSR2 polega na okresowym włączeniu i wyłączeniu blokady reakcji na sygnały zewnętrzne (użyto polecenia SIGPROC)