Zadanie A
Prosty shell.

Należy zaimplementować shell, który wykonuje polecenia odczytywane ze standardowego wejścia.

Każde linia standardowego wejścia może zawierać do 10 poleceń połączonych pipe'ami. Każde polecenie może mieć co najwyżej 10 argumentów. Można przyjąć, że długości linii nie przekraczają 2048 znaków. Można przyjąć, że polecenia i argumenty nie zawierają białych znaków. Do parsowania linii poleceń można wykorzystać dostarczony kod (z ćwiczeń).

Polecenie składa się z:

1. ścieżki do pliku uruchamialnego / nazwy komendy shella,
2. argumentów,
3. przekierowania wejścia / wyjścia.

Ponadto linia poleceń może zawierać na końcu symbol & oznaczający, że wszystkie polecenia z linii powinny byc wykonane w tle. Jeśli linia zawiera więcej niż jedno polecenie to są one oddzielone symbolem "|".

1. Uruchamianie poleceń. Każde polecenie, które nie jest komendą shella powinno być wykonywane w procesie potomnym. Jeśli nie jest wykonywane w tle to shell powinien zawiesić swoje działanie aż do zakończenia procesu.
2. Komendy. Niektóre polecenia mają być wykonywane przez proces shella. Należy zaimplementować:
   - `exit` - kończy proces shella
   - `cd`
   - `echo`
   - `kill signum pid`
   - `lenv` - wypisuje na wyjście zmienne środowiskowe procesu shella. Uwaga przekierowania powinny działać również dla tej komendy.

   Można przyjąć, że komendy występują zawsze jako jedyne polecenie w linii (tzn nie są łączone pipe'ami).
3. Uruchamianie procesów w tle. Jeśli linia poleceń jest zakończona znakiem & to wszystkie polecenia z linii mają być wykonane w tle. Oznacza to że proces shella

powinien od razu przetwarzać kolejne polecenia, nie czekając na zakończenie procesów dotyczących poleceń z bieżącej linii. Należy zadbać o eliminacje procesów zombie. Jeśli proces w tle się zakończył, proces shella powinien poinformować o tym użytkownika przed wczytaniem kolejnej linii (np. przed wypisaniem prompta), wystarczy wypisać numer zakończonego procesu i jego status.

4. Przekierowania strumieni. Należy zaimplementować przekierowania >, >> , <. Przekierowania mają wyższy priorytet niż pipe'y.

5. Łączenie procesów za pomocą `pipe`. Jeśli linia poleceń zawiera więcej niż jedno polecenie to muszą być oddzielone znakiem |. Jeśli w linii występują polecenia $p_1, p_2, \ldots$ to procesy wykonujące kolejne polecenia powinny być połączone w taki sposób, że standardowe wyjście procesu polecenia p_1 powinno być standardowym wejściem procesu polecenia p_2 itd. Wyjątkiem jest sytuacja, kiedy polecenia składowe specyfikują przekierowania z/do plików. W przypadku poleceń połączonych pipe'ami, które nie są wykonywane w tle, wystarczy żeby shell czekał na zakończenie ostatniego z nich.

6. Obsługa sygnałów. Proces shella powinien obsługiwać sygnały SIGCHLD. Należy zadbać o to, żeby sygnał SIGINT (wprowadzony z terminala CTRL-C) nie przerywał działania shella, ani procesów działających w tle. Powinien natomiast przerywać działanie procesu/procesów uruchamianych normalnie (na które shell czeka).

W razie wątpliwości proszę się przyjrzeć jak działa `ash` (domyślny shell w MINIX'ie).

Dodatkowe wymagania:

- Shell powinien rozpoznać czy standardowemu wejściu odpowiada specjalne urządzenie znakowe i tylko w takim przypadku wypisywać na stdout prompt i informacje o zakończonych procesach z tła.
- Do czytania ze standardowego wejścia należy używać funkcji `read`.
- Program powinien kompilować się i działać w domyślnej konfiguracji MINIX'a w wersji 3.1.0 (book version).
- Należy wysłać jedno archiwum `tar` zawierające pliki źródłowe oraz `Makefile`.