Wprowadzenie do środowiska PVM

Zakres ćwiczenia

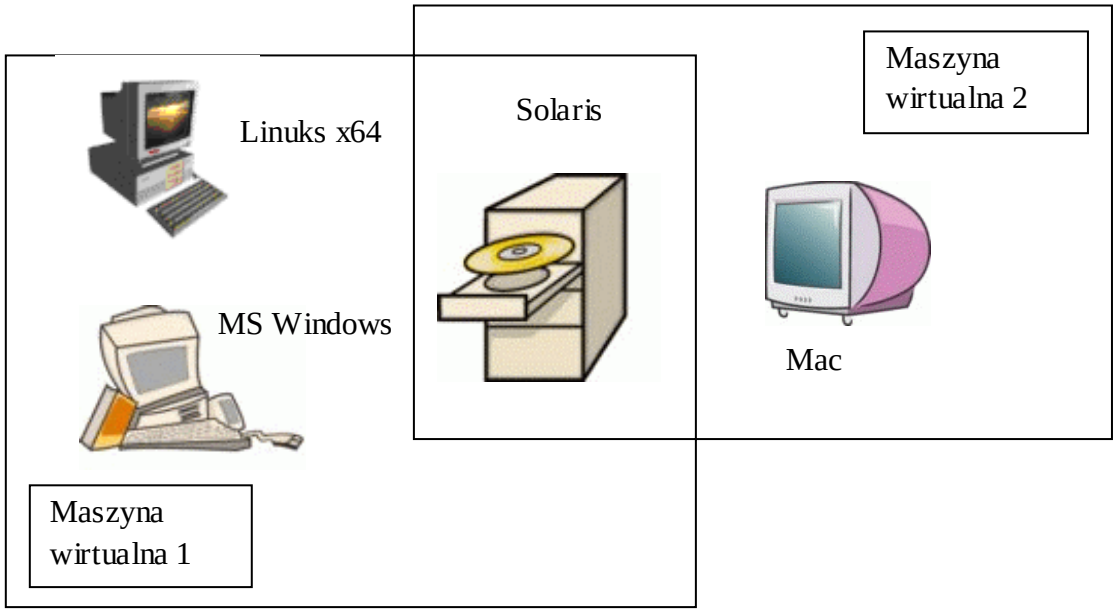
W tym ćwiczeniu zainstalujesz i skonfigurujesz do dalszej pracy środowisko PVM. Poznasz dwa polecenia konsoli PVM oraz zapoznasz się z historią i podstawowymi elementami tego środowiska.

Historia środowiska PVM

PVM (ang. *Parallel Virtual Machine*) powstał w 1989 roku w Oak Ridge National Laboratory, w celu ułatwienia programowania równoległego w środowisku komputerów heterogenicznych połączonych siecią. Autorami systemu byli Vaidy Sunderam oraz Al. Geist. Pierwsza wersja środowiska nie była dystrybuowana, używano ją jedynie wewnętrznie. Kolejna wersja została napisana na uczelnie University of Tennessee w 1991. Po szerokim odzewie od użytkowników wydano kilka kolejnych modyfikacji środowisko (do wersji 2.4 włącznie). W 1993 wydano wersję trzecią środowiska, przy czym zmiana numeru odzwierciedlała fakt jego kompletnego przepisania od podstaw. Obecnie środowisko PVM nie jest już rozwijane równie dynamicznie jak kiedyś, lecz wciąż jeszcze jest stosunkowo szeroko używane (zwłaszcza w kręgach akademickich) chociaż można zaryzykować stwierdzenie, że prymat utraciło na rzecz środowiska MPI.

Podstawowe elementy środowiska PVM

Idea PVM bazuje na pojęciu maszyny wirtualnej, czyli pewnej dodatkowej warstwy oprogramowania, na poziomie której poszczególne heterogeniczne węzły systemu (ang. *hosts*) postrzegane są jako jednostki przetwarzające, a cała maszyna wirtualna jest postrzegana jako komputer równoległy z rozproszoną pamięcią. Maszyn wirtualna konfigurowana jest dla poszczególnych użytkowników, co oznacza, że ten sam węzeł może być składnikiem wielu maszyn wirtualnych. Inaczej rzecz ujmując, różni użytkownicy tego samego komputera (hosta) mogą jednocześnie włączyć go do swoich maszyn wirtualnych (Rysunek 1). Podstawowym elementem konstrukcyjnym maszyny wirtualnej jest demon systemowy pvmd. Każdej maszynie wirtualnej, w skład której wchodzi dany węzeł, odpowiada na tym węźle jeden demon.



Rysunek 1 Koncepcja maszyny wirtualnej

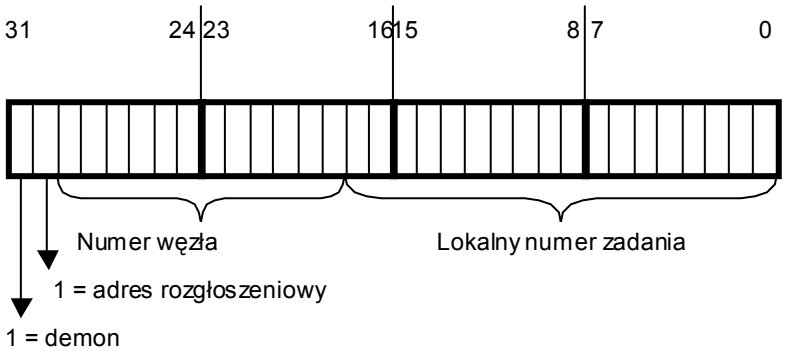
Przetwarzanie w środowisku PVM polega na wykonywaniu programów tworzących aplikację równoległą na poszczególnych węzłach maszyny wirtualnej. Na poziomie maszyny wirtualnej elementarną jednostką reprezentującą wykonywany program jest *zadanie* (ang. *task*). Każde zadanie jest też procesem w lokalnym systemie operacyjnym węzła, a jego praca jako zadania PVM jest kontrolowana przez lokalny demon w tym węźle. Ponieważ lokalny demon również jest zadaniem PVM, zadania realizujące program aplikacji będą nazywane zadaniami użytkowymi.

Podstawową funkcją środowiska PVM jest ułatwianie uruchamiania zadań na poszczególnych węzłach maszyny wirtualnej oraz dostarczanie mechanizmów do komunikacji, synchronizacji i identyfikacji tych zadań niezależnie od ich fizycznej lokalizacji na poszczególnych węzłach. Ponadto, PVM umożliwia dynamiczną zmianę konfiguracji maszyny wirtualnej oraz udostępnia mechanizm dynamicznych grup procesów, ułatwiających implementację algorytmów równoległych. Skorzystanie z funkcjonalności PVM wymaga użycia w programach zadań funkcji PVM, których implementacja jest dołączana na etapie konsolidacji w postaci biblioteki *libpvm3.a* i *libgpvm3.a* dla programów w języku C lub *libfpvm3.a* dla programów w języku Fortran

Wybrane elementy konstrukcji środowiska PVM

Większość decyzji projektowych podjętych przez twórców środowiska PVM wynikała z założenia, że powinno być ono dostępne na możliwie dużej liczbie platform sprzętowych i systemowych. Podstawowym systemem operacyjnym, na którym działał PVM był UNIX w niemal wszystkich jego odmianach. Z czasem pojawiły się realizacja na VMS oraz Microsoft Windows. Pewne rozwiązania przyjęto zatem z względu na popularność i szeroką dostępność wykorzystanych mechanizmów, często koszem efektywności.

Kluczowym elementem w systemie rozproszonym jest komunikacja pomiędzy zadaniami, która wymaga z kolei mechanizmu jednoznacznej identyfikacji zadań oraz ich fizycznej lokalizacji w systemie. W PVM zadania identyfikowane są przez 32-bitową wartość liczbową, zwaną *TID* (ang. *task identifier*), nadawaną przez lokalny demon. Wartość tego identyfikatora składa się z czterech pól, zawierających odpowiednio: numer węzła, na którym działa zadanie, numer zadania na tym węźle, bit identyfikujący zadanie-demon i bit określający adres rozgłoszeniowy (ang. *multicast address*). Format identyfikator przedstawia Rysunek 2.



Rysunek 2 Identyfikator zadania

W środowisku PVM w ogólności przyjmuje się, że zadania nie mają możliwości współdzielenia jakiegokolwiek obszaru pamięci, zatem kooperacja między nimi odbywa się tylko poprzez przekazywanie komunikatów (ang. *message passing*). Komunikat identyfikowany jest przez TID nadawcy i przez etykietę, tj. 16-bitową liczbę całkowitą. Model komunikacyjny przyjęty w PVM zakłada, że każde zadanie może wysyłać komunikaty do każdego innego, oraz że nie istnieją ograniczenia na rozmiar i ilość komunikatów. Jedynym ograniczeniem jest fizyczny rozmiar pamięci na poszczególnych węzłach maszyny wirtualnej.

PVM zapewnia komunikację asynchroniczną, co oznacza, że po wysłaniu komunikatu zadanie-nadawca nie czeka, aż komunikat dotrze do odbiorcy, tylko natychmiast kontynuuje swoje przetwarzanie. Odpowiedzialność za przekazanie komunikatu przejmują odpowiednie demony. Maszyna wirtualna gwarantuje jedynie zachowanie lokalnego porządku (porządku FIFO), tzn. jeśli

jedno zadanie wysyła dwa komunikatu do innego zadania, to komunikaty zostaną doręczone w kolejności, w jakiej zostały nadane. Adresatem komunikatu może być pojedynczy proces identyfikowany przez konkretny TID, może to być zbiór procesów, których identyfikatory przekazywane są w odpowiedniej strukturze, lub może to być dynamiczna grupa procesów identyfikowanych przez nazwę.

Odbiór komunikatu polega na jego wyciągnięciu z bufora po stronie odbiorczej, przy czym możliwe są dwie formy odbioru: blokująca i nieblokująca. Odbiór blokujący oznacza, że zadanie odbierające zostaje zawieszone do momentu dotarcia komunikatu, jeśli nie dotarł on wcześniej. W przypadku odbioru nieblokującego funkcja zwraca sterowanie natychmiast, przekazując odpowiedni status zależnie od tego, czy komunikat dotarł czy nie.

Komunikacja pomiędzy zadaniami najczęściej odbywa się za pośrednictwem demonów, które kontrolują pracę tych zadań. W komunikacji pomiędzy demonami wykorzystywany jest protokół UDP, a komunikacja pomiędzy zadaniem a lokalnym demonem odbywa się na protokole TCP. Użycie protokołu UDP w komunikacji pomiędzy demonami daję większą elastyczność, ale ze względu na zawodność wymaga dobudowania dodatkowej funkcjonalności w postaci mechanizmu potwierdzeń i retransmisji.

Poszczególne zadania aplikacyjne uruchamiane są przez odpowiednie demony. W czasie konfiguracji maszyny wirtualnej muszą zostać również zdalnie uruchomione same demony. W tym celu wykorzystywana jest usługa rsh (ang. *remote shell*), umożliwiająca zdalne wykonanie polecenia bez potrzeby logowania użytkownika w trybie interaktywnym. Po odpowiednim skonfigurowaniu pliku `~/.rhosts` uruchamianie zdalnych poleceń możne się odbywać bez podawania hasła.

Elementem ułatwiającym programowanie równoległe poprzez możliwość dynamicznej dekompozycji instancji problemu jest mechanizm grupowania zadań. Zarządzanie grupami zadań w środowisku PVM jest całkowicie scentralizowane i spoczywa na serwerze grup . pvmgs. Jest to specjalne zadanie systemowe, uruchamiane automatycznie na każdej maszynie wirtualnej . dokładniej na pierwszym węźle tej maszyny w momencie utworzenia pierwszej grupy procesów. Z zadaniem tym komunikują się inne zadania lub demony w czasie wykonywania funkcji grupowych. Rozwiązanie scentralizowane ma istotną wadę - istnieje ryzyko powstania wąskiego gardła w przypadku intensywnego korzystania z mechanizmu grup przez zadania działające na maszynie wirtualnej.

Instalacja i przygotowanie środowiska PVM

W celu rozwijania programów PVM oczywiście należy na początku przygotować sobie środowisko pracy, instalując i konfigurując PVM na każdej z maszyn, która ma stać się węzłem maszyny wirtualnej. Instalacja środowiska różni się w zależności od rodzaju systemu operacyjnego. Do instalacji wymagane są:

- Zainstalowany system Linuks z zestawem pakietów dla programisty (kompilator gcc) oraz pakiety rsh lub ssh (zalecane). Kurs przygotowywany jest w oparciu o dystrybucję SuSe Linuksa.
- System Windows (nie zalecane) z programem WinRSH oraz narzędzia dla programisty (kompilatory).

Student powinien posiadać wiedzę odpowiednią do konfiguracji rsh lub ssh oraz podstawową znajomość administracji wybranym systemem operacyjnym. Środowisko pracy powinno zostać przygotowane do pracy w sieci.

Instalacja w systemie Linuks

Najprostsza jest ona w przypadku Linuksa – zazwyczaj istnieją przygotowane pakiety dla każdej ważniejszej dystrybucji Linuksa. Tak więc instalacja PVM sprowadza się do wydania odpowiedniego polecenia (różniącego się w zależności od typu pakietów używanego w dystrybucji). Pakiety te zazwyczaj są dostępne razem z dystrybucją (na płytkach instalacyjnych) albo w jednym z licznym repozytoriów oprogramowania. W systemach linuksowych opartych o pakiety RPM, na przykład SuSe, wystarczy wydać następujące polecenia (instalujące, odpowiednio, podstawową część środowiska wystarczającą do uruchomienia gotowych programów oraz pliki niezbędne dla programistów do rozwijania nowych aplikacji):

```
root@linuxlab/~# rpm -i pvm-3.4.5-6.x86_64.rpm
root@linuxlab/~# rpm -i pvm-devel-3.4.5-6.x86_64.rpm
```

Następnie należy środowisko skonfigurować. Sprowadza się to do ustawienia wartości kilku zmiennych środowiskowych. Odpowiednie polecenia najlepiej wpisać również do plików startowych specyficznych dla używanej powłoki (np. `.bashrc` albo `.profile` dla powłoki `bash`), tak by nie musieć powtarzać tej czynności przy każdym uruchomieniu nowej powłoki):

- Zmienna PVM_ROOT wskazuje na lokalizację środowiska PVM. Zazwyczaj jest to `/usr/lib/pvm3` (należy koniecznie ustawić tą zmienną!);
- Zmienna PVM_ARCH określa architekturę danego węzła. Może to być `LINUX`, `LINUX64` itd.
- Zawartość zmiennej PVM_HOME to nazwa katalogu, w którym domyślnie będą poszukiwane programy PVM

Należy również uzupełnić ścieżkę przeszukiwań, tak by zawierała ona katalogi z plikami uruchomieniowymi środowiska PVM.

```
export PVM_ROOT=/usr/lib/pvm3
export PVM_ARCH=`pvmgetarch`
export PVM_HOME=$HOME/pvm3/bin/$PVM_ARCH
export PVM_SRC=$HOME/pvm3/src
export PATH=$PATH:$PVM_ROOT/bin:$PVM_ROOT/lib:$PVM_HOME
```

Podczas uruchomienia środowiska demon PVM będzie się starał (sposób uruchomienia PVM oraz dodawania węzłów zostanie omówiony w dalszej części ćwiczenia) zdalnie uruchomić demonów PVM na wszystkich węzłach maszyny wirtualnej. Domyślnie używany jest do tego program `rsh`, który wymaga oczywiście uruchomienia demona `rshd` odpowiedniego przygotowania plików `.rhosts` na każdym z węzłów.

```
nazwa_węzła1 nazwa_użytkownika
nazwa_węzła2 nazwa_użytkownika
```

Rozwiązaniem bezpieczniejszym jest użycie programu `ssh`. Oczywiście na każdym węźle musi wtedy działać demon `sshd` (zazwyczaj tak jest w wielu dystrybucjach w konfiguracji domyślnej). Na przykład,

```
root:~#chkconfig sshd
sshd  off

root:~#insserv sshd
```

w dystrybucji SuSe wykonuje się to za pomocą następujących poleceń:

Wybór `ssh` dokonywany jest dzięki ustawieniu zmiennej PVM_RSH.

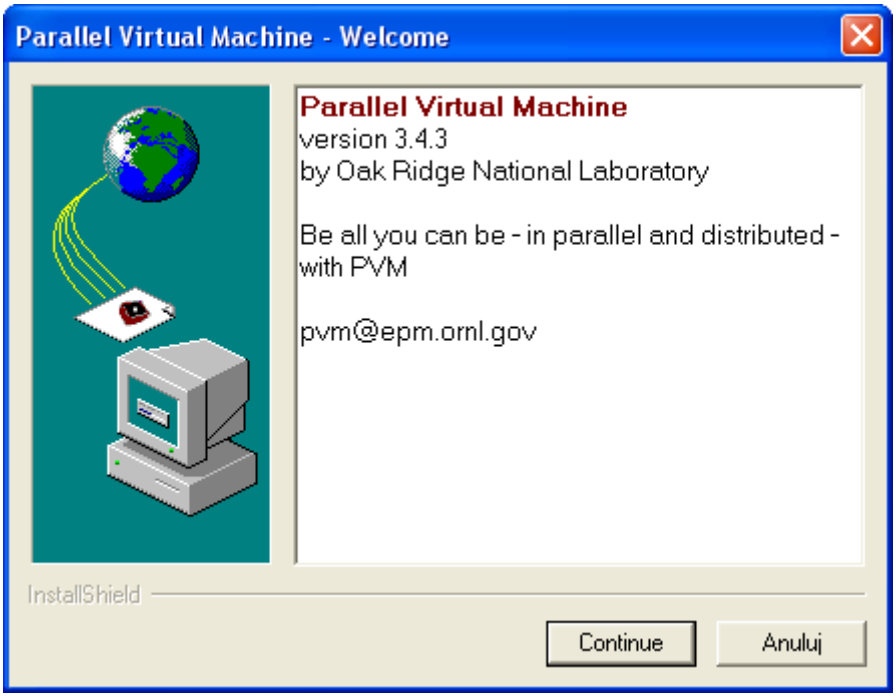
```
export PVM_RSH=/usr/bin/ssh
```

W tym momencie środowisko jest przygotowane do pracy.

Instalacja w systemie Windows

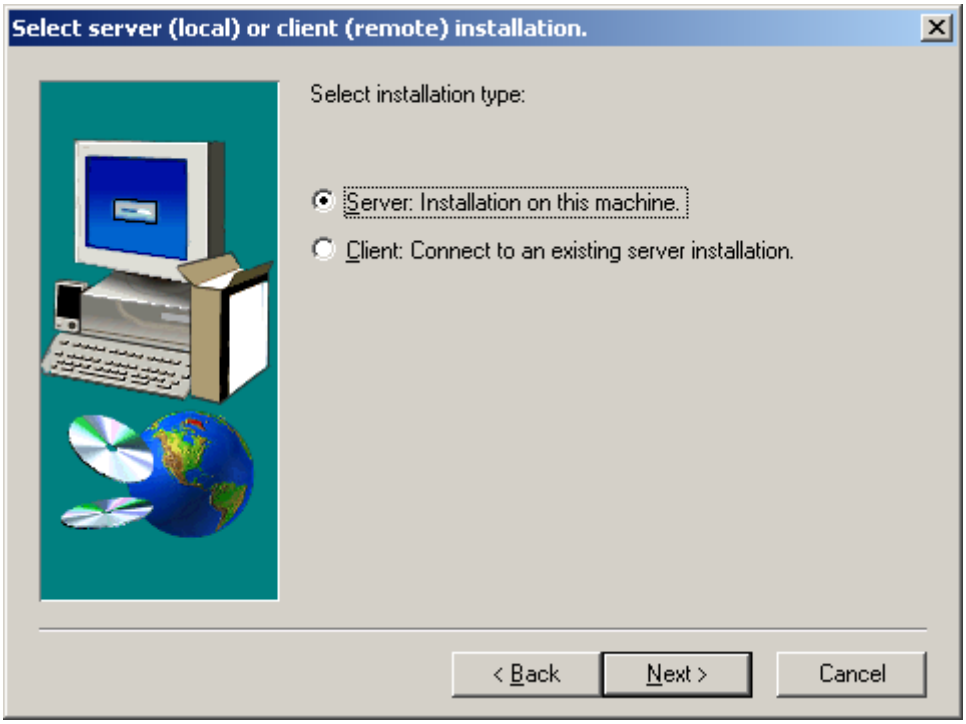
Istnieje możliwość zainstalowania PVM w systemie Windows, jednakże jest to nie zalecane na potrzeby kursu, gdyż ćwiczenia zostały przygotowane z użyciem Linuksa. W związku z tym, ewentualne problemy pojawiające się w czasie instalacji w tym systemie studenci będą musieli

rozwiązywać na własną rękę. W dalszej części kursu będziemy zakładać, że PVM działa w systemie Linuks. Gotowy program instalacyjny PVM można znaleźć pod adresem http://www.csm.ornl.gov/pvm/pvm_home.html. Instalacja w tym wypadku polega na uruchomieniu programu instalacyjnego (Rysunek 3)



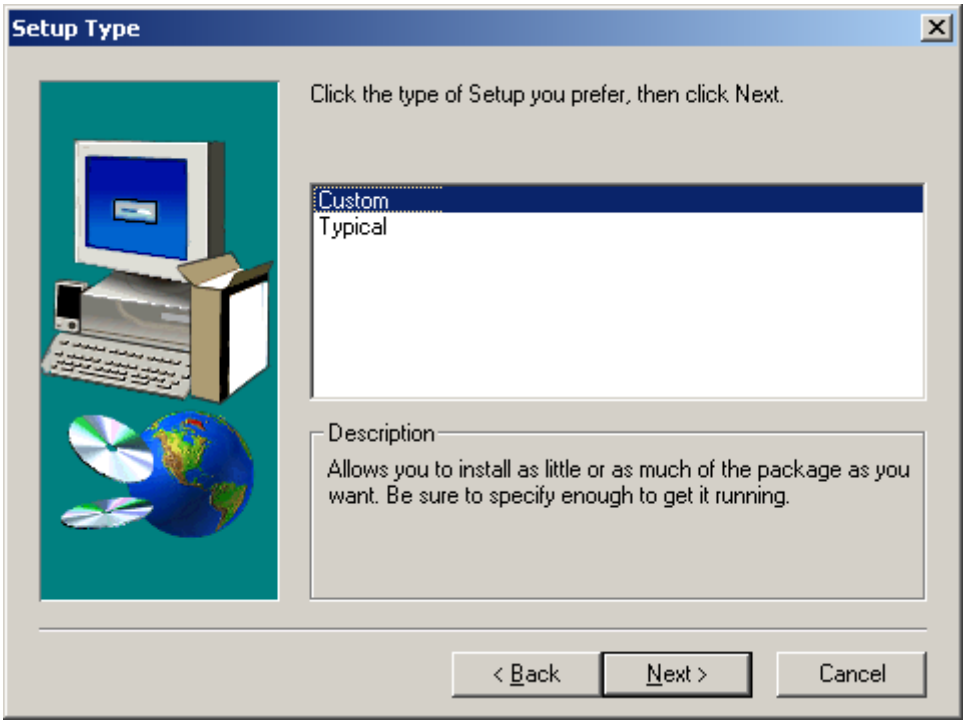
Rysunek 3 Ekran początkowy instalatora środowiska PVM dla systemu Windows

Na kolejnych ekranach instalatora, po wyrażeniu zgody na warunki licencji i przejrzeniu informacji na temat środowiska, wystarczy wybrać instalację w trybie serwera.

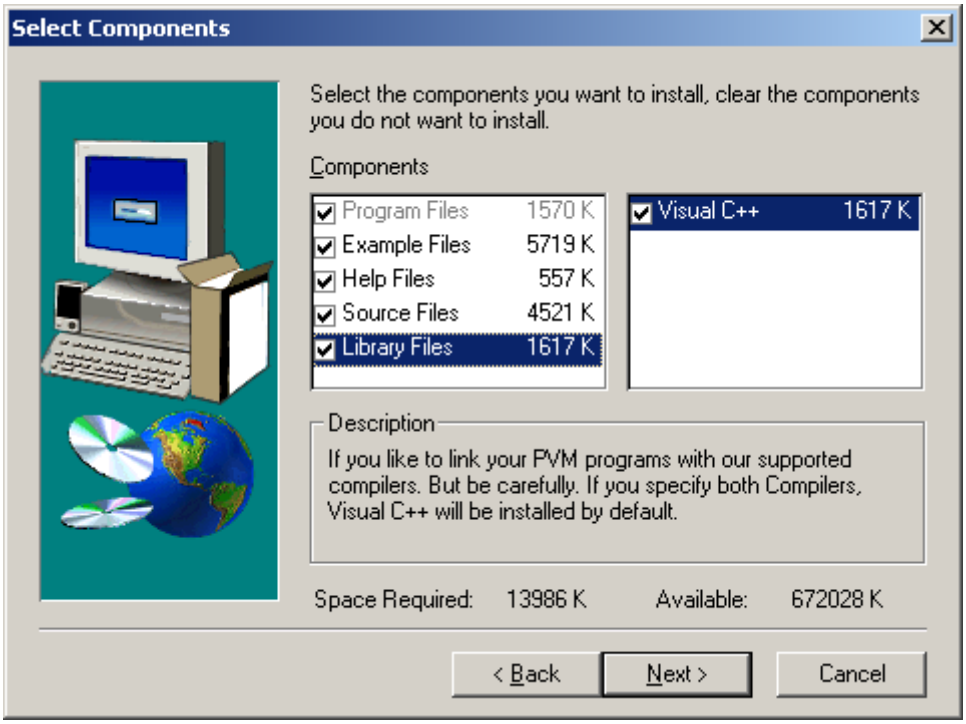


Rysunek 4 Instalacja środowiska PVM

W dalszych krokach należy wybrać lokalizację dla instalowanego środowiska, po czym należy wybrać opcję „Custom” i zaznaczyć wszystkie możliwe opcje.



Rysunek 5 Wybór rodzaju instalacji środowiska PVM



Rysunek 6 Składniki instalowanego środowiska PVM

Po zakończeniu instalacji dobrze jest się upewnić, czy odpowiednie zmienne środowiskowe (PVM_ROOT, PVM_ARCH, PVM_TMP) zostały ustawione przez instalatora, i dokonać ich ewentualnej korekty (W systemach Windows 2000/XP za pomocą Panel Sterowania->System-

>Zakładka zaawansowane->Zmienne środowiskowe). Ważne jest tutaj, by zmienne środowiskowe, np. PVM_TMP nie były dłuższe niż 32 znaki – wielu użytkowników zgłaszało problemy, jeżeli ten warunek nie był spełniony. Dobrze jest także, jeżeli zmienne nie zawierają znaków spacji.

Przed dodaniem nowych węzłów do maszyny wirtualnej należy skonfigurować demony `rexec` i `rsh`, dostępne na przykład z adresu *http://www.winrshd.com*.

Weryfikacja poprawności instalacji

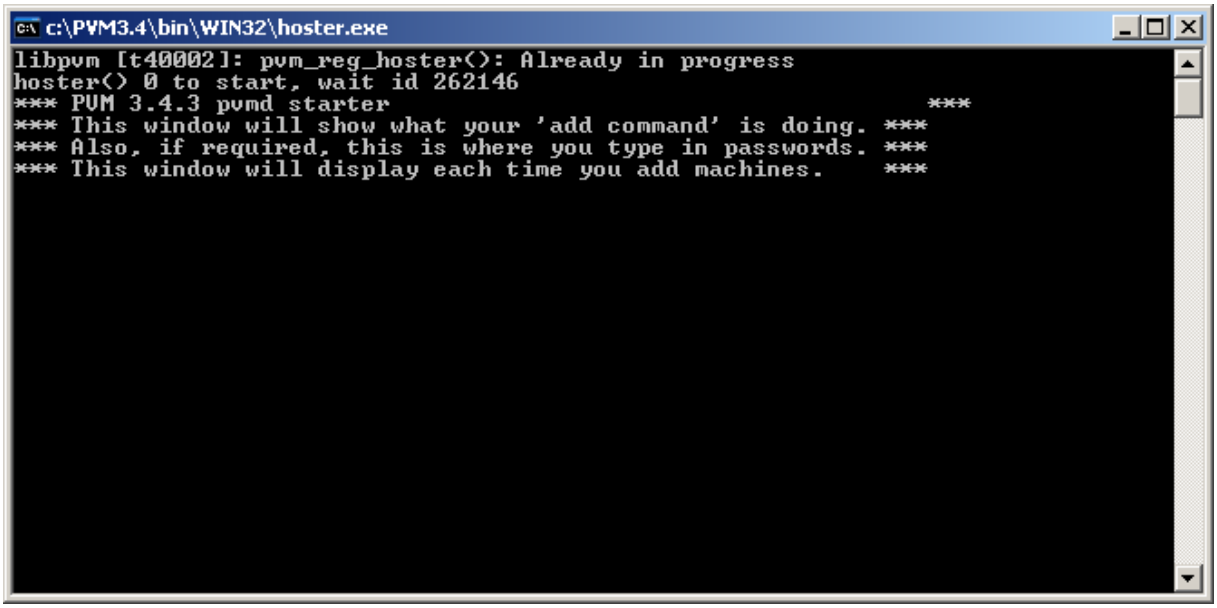
Po zainstalowaniu i skonfigurowaniu środowiska można je uruchomić. W systemie Linuks wystarczy wydać polecenie `pvm`, w wyniku którego zostanie uruchomiona konsola pvm. Konsola PVM jest zadaniem systemowym, które udostępnia wybrane polecenia PVM w trybie interaktywnym, tzn. można wydać polecenie interpretowane przez konsolę PVM, i natychmiast zobaczyć wyniki. Konsola udostępnia głownie funkcje związane z obsługą zadań i konfiguracją maszyny wirtualnej i najczęściej wykorzystywana jest do dynamicznej zmiany konfiguracji maszyny wirtualnej oraz do kontroli pracy zadań. W przypadku poprawnego działania powinien pojawić się znak zachęty `pvm>` bez żadnych dodatkowych komunikatów. Wyjście z konsoli dokonuje się za pomocą wydania polecenia konsoli PVM `quit`; sam demon pvm dalej działa (komunikat `pvmd still running`), i ponowne wydanie polecenia `pvm` spowoduje tylko uruchomienie konsoli (pojawia się wtedy komunikat `pvmd already running`). Aby zatrzymać środowisko PVM należy wydać polecenie `halt` w konsoli PVM (zakończenie tego polecenia sygnalizowane jest komunikatem `Terminated`) – to polecenie należy wydawać pod koniec każdej sesji z PVM.

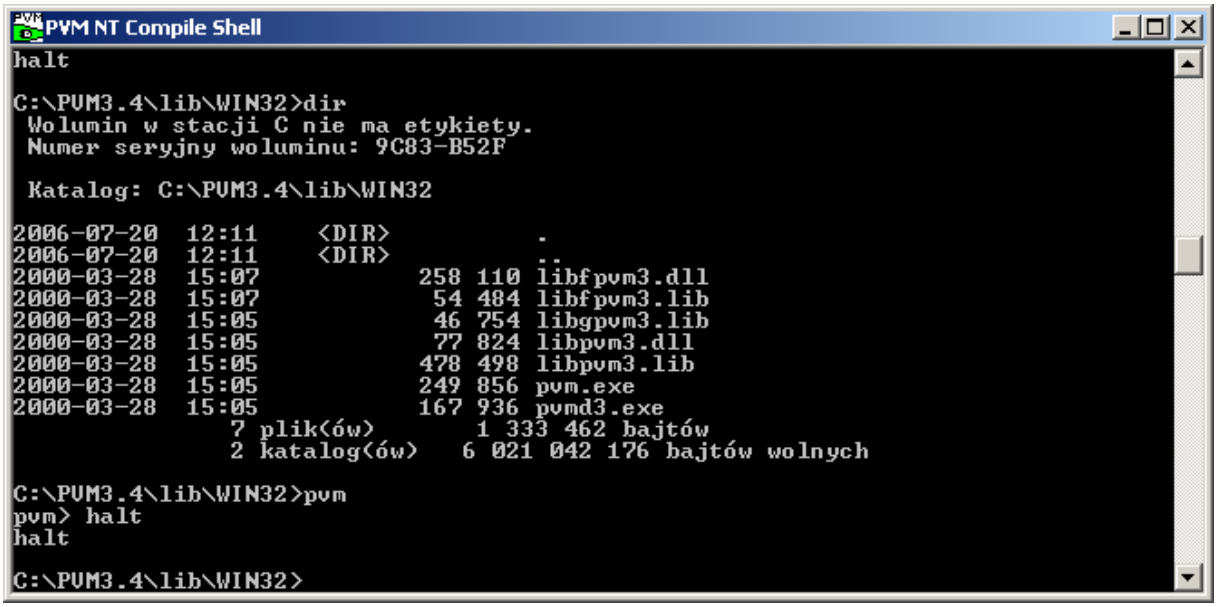
```
user@linuxhost:~> pvm
pvm> quit

Console: exit handler called
pvmd still running.
user@linuxhost:~>
```

```
user@linuxhost:~>pvm
pvmd already running.
pvm> halt
Terminated
user@linuxhost:~>
```

W przypadku błędnego zakończenia środowiska PVM (na przykład, przypadkowemu wyłączeniu komputera przed wydaniem polecenia `halt`) w celu uniknięcia trudnych do zdiagnozowania błędów należy wyczyścić zawartość katalogu tymczasowego `/tmp`.

W systemie Windows należy uruchomić konsolę za pomocą Menu Start->Programy->PVM3.4->PVM Console (o ile lokalizacja ta nie została zmieniona w czasie instalacji) lub za pomocą polecenia `pvm` (a dokładniej mówiąc <lokalizacja pvm>\lib\win32\pvm). Rysunek 7 poniżej przedstawia efekt uruchomienia środowiska PVM w systemie Windows. Oprócz znaku zachęty PVM pojawia się krótko dodatkowe okienko:





Rysunek 7 Efekt uruchomienia konsoli PVM w systemie Windows

W dalszej części kursu zakładać będziemy pracę w systemie Linuks.

Podsumowanie

Podczas tego ćwiczenia poznałeś historię oraz ogólną budowę środowiska PVM. Powinieneś posiadać działające, zainstalowane i skonfigurowane do pracy środowisko PVM.

Co powinieneś wiedzieć:

- Jak uruchamiać środowisko PVM oraz jak uruchomić konsolę (polecenie `pvm`)
- Jak wychodzić z konsoli bez kończenia pracy z środowiskiem PVM (polecenie `quit` konsoli PVM)
- Jak prawidłowo kończyć pracę z środowiskiem PVM (polecenie `halt` konsoli PVM)