

Przygotowanie środowiska kompilacji śpiewników i nut

Andrzej Stanisław Odyniec

11 marca 2023

Niniejsze informacje są tutaj zebrane na wypadek, gdyby ktoś chciał przejąć całość aktualizacji śpiewników i nut, szczególnie wtedy, gdy ja już nie będę mógł tego robić. Wprawdzie paczki dystrybucyjne ze źródłami zawierają wystarczający zestaw, by skompilować tak śpiewnik jak i nuty, jednak nie zawierają całego systemu automatyzacji odbudowy zestawu (np. mechanizmu synchronizacji z serwerem). Także zestaw narzędzi do kompilacji, chociaż jest oprogramowaniem wolnym, wymaga pobierania ze świata i poinstalowania. Na dodatek jest to oprogramowanie żywe i nie każda wersja każdego komponentu dobrze współpracuje z pozostałymi. Ten opis nie rozwiązuje problemu konieczności poznania $\text{T}_{\text{E}}\text{X}$ a, $\text{L}_{\text{A}}\text{T}_{\text{E}}\text{X}$ a, zasad muzyki, notacji nutowej, systemów komputerowego składu nut i szeregu innych spraw związanych z informatyką, typografią, muzyką. Niemniej ma za zadanie pomóc zbudować środowisko do $\text{T}_{\text{E}}\text{X}$ owania śpiewników oraz nut. Wszystkie elementy opisu zostały sprawdzone na maszynie wirtualnej z Windows 11 postawionym od zera.

Potrzebujemy komputera z systemem Windows 10 lub Windows 11 w wersji 64-bit. W wersji 32-bit zapewne też by zadziałało, ale nie dysponuję wersją 32-bit $\text{MiK}_{\text{T}}\text{E}_{\text{X}}$ a z roku 2018 a z aktualnym to trzeba będzie ponownie uruchamiać skład nut. Możemy też skorzystać z maszyny wirtualnej. Testowałem to pod **VMware workstation** ale zapewne może to być też windowsowe **Hyper-V**. Obawiam się, że nie może to być **Piaskownica**. Pomiędzy Windows 10 a Windows 11 są pewne, niewielkie różnice, na które trzeba zważać.

1. Instalujemy $\text{MiK}_{\text{T}}\text{E}_{\text{X}}$ a. Tutaj ważne ostrzeżenie! Pełna instalacja $\text{MiK}_{\text{T}}\text{E}_{\text{X}}$ a waży prawie 3GB i jej pobranie musi nieco trwać. W zależności od łącza, może to być nawet kilkadziesiąt minut.

Można pomagać się z najnowszą dystrybucją $\text{MiK}_{\text{T}}\text{E}_{\text{X}}$ a, pobraną przez instalator ze strony <https://miktex.org/download> przy pomocy Instalatora sieciowego 64-bit. Najrozsądniej pobierać wszystko z serwera <http://sunsite.icm.edu.pl/packages/tex/systems/win32/miktex/tm/packages/>. Trzeba się jednak liczyć z dodatkową pracą przy adaptacji źródeł. Np. po drodze zmieniono silnik i wywołanie pakietu **polyglossia** w $\text{T}_{\text{E}}\text{X}$ u wymaga raczej wskazania innego silnika. Coś się po drodze działo z pakietami obsługującymi preprocessing nut i nuty mogą się tak od razu dobrze nie składać.

Zdecydowanie stabilna jest wersja z marca 2018, której używam, przynajmniej w zakresie składu nut. Dlatego proponuję instalację z zachowanego przeze mnie archiwum (znajdującego się w paczce **Environment.zip**). W nim znajdziemy plik **setup** w wersji **setup-2.9.6637-x64.exe** oraz katalog z pakietami: **MiKTeX 2.9 64 Setup**. Aby zainstalować to archiwum, należy:

- a) umieścić (rozpakować) całość (albo z *niniejszego hiperłącza* albo patrz na końcu, gdzie paczka jest) w jakimś stabilnym miejscu; ja to robię zwykle w jakimś podkatalogu w drzewie katalogów **c:\Install**
- b) uruchomić **setup-2.9.6637-x64.exe** i zaakceptować warunki licencji, pójść Dalej
- c) wybrać operację **Install MiKTeX** (bo archiwum zostało już przeze mnie ściągnięte w roku 2018), pójść Dalej
- d) wybrać **Complete MiKTeX**, pójść Dalej
- e) wybrać tryb instalacji (najlepiej dla wszystkich użytkowników), pójść Dalej
- f) w polu **Install MiKTeX from:** wpisać (lub wybrać przyciskiem **Browse...**) ścieżkę, gdzie umieściliśmy archiwum pakietów, pójść Dalej

- g) zaakceptować docelową ścieżkę instalacji (`C:\Program Files\MiKTeX 2.9`) i Dalej
 - h) zaakceptować papier A4 i zdecydować, czy będziemy zatwierdzać brakujące pakiety czy instalować w locie bez pytania (opcja Yes)? Te ustawienia potem będzie można zmienić z konsoli MiKTeXa; idziemy Dalej
 - i) teraz możemy nacisnąć **Start** i pozwolić na wprowadzanie zmian w komputerze.
 - j) Instalacja potrzebuje 15+ minut.
2. Jeżeli zechcemy składać innymi fontami, np. rodziną Minion/Myriad/Caflish Script, będziemy potrzebowali tych fontów w systemie.
- W systemie Windows 11 należy wybrać procedurę instalacji fontów dla wszystkich użytkowników a nie tylko dla aktualnego (co jest domniemywane). Fonty instalowane dla wszystkich użytkowników umieszczane są w katalogu systemowym `C:\Windows\Fonts`, gdy tymczasem fonty instalowane dla aktualnego użytkownika są umieszczane w katalogu lokalnym `C:\Users\<nazwa-użytkownika>\AppData\Local\Microsoft\Windows\Fonts` i standardowo LuaTeX (przynajmniej ten z roku 2018) tego katalogu w poszukiwaniu fontów nie przegląda. Instalację dodatkowych fontów lepiej zrobić przed pierwszym użyciem LuaTeXa, gdyż podczas pierwszego użycia zbuduje on bazę fontów. Późniejsze instalacje fontów wymagają przebudowywania bazy fontów.
3. Jeżeli będziemy składać także nuty śpiewnikowe, znajdzie potrzeba zainstalowania wolnego fontu **Noto Sans Hebrew**, gdyż w komentarzach do nut występują niekiedy odwołania do tytułów pieśni hebrajskich, których szuka się w sieci i na YouTube pod tytułami zapisanymi alfabetem hebrajskim (pisanym i czytanim od prawej do lewej). Font instalujemy podobnie jak inne dodatkowe fonty (w Windows 11 dla wszystkich użytkowników). Plik tego fontu `NotoSansHebrew-Regular.ttf` jest umieszczany wraz ze źródłami nut śpiewnikowych w paczce `NutySpiewowNeokatechumenalnychSrc.zip`. Można go też pobrać z pakietem **Noto Sans** na stronie fontu <https://fonts.google.com/specimen/Noto%20Sans> (są tam cztery odmiany). Można wreszcie pobrać komplet fontów Google’a przy pomocy URL: <https://github.com/google/fonts/archive/main.zip> (uwaga, około 970MB!), ale tutaj są dostępne tylko fonty zmienne z parametrami sterowanymi z CSS API i ewentualne odmiany trzeba byłoby czymś wygenerować.
4. Możemy zainstalować edytor **TeXStudio**; Nie jest niezbędny do kompilacji, ale pomaga edytować i testować pliki w **TeXu** i **LaTeXu**; tę instalację lepiej wykonać po zainstalowaniu **MiKTeXa**, bo wtedy sama się skonfiguruje pod kątem jego lokalizacji.
- a) Instalację pobieramy z <https://www.texstudio.org/>
 - b) Po zainstalowaniu wchodzimy w **Opcje | Konfiguruj** i w grupie **Zbuduj** ustawiamy kompilator domyślny na **LuaLaTeX** a w **Sprawdzanie języka | Domyślny język** ustawić na **PL**.
5. Niezbędne będą programy linuxowe typu GnuWin32, choćby ze względu na AWKa. Sam GnuWin32 jest od dawna nieaktualizowany i chociażby AWK z tego zbioru nie obsługuje pewnych funkcji. Dlatego potrzebne pakiety należy pozbierać i zainstalować ręcznie. Programy te wspomagają korekty oraz sortowanie plików z indeksami i są wywołane pomiędzy przebiegami **TeXowania** (nuty liturgiczne nie zawierają sortowanych indeksów, ale reszta owszem).
- a) Programy użytkowe wiersza poleceń używane w skryptach, jak **grep** czy **ls**, wygodne do przetwarzania plików w katalogu lub wyszukiwania czegoś w plikach czy listingach, można pobrać z pakietu **MinGW**.
Ze względu na różnorodność użytków i zależności, aby pobierać programy **MinGW**, potrzebny nam będzie instalator i menedżer systemu. W celu jego zainstalowania wchodzimy na stronę projektu **MinGW** <https://osdn.net/projects/mingw/releases/68260> i pobieramy program `mingw-get-setup.exe` a następnie uruchamiamy go. Program ten ściągnie instalator pakietów linuxowych dla Windows z pakietu **MinGW** o nazwie **MinGW Installation Manager Setup tool**, zainstaluje go i uruchomi.

W tym menedżerze będziemy wybierać odpowiednie pakiety, pobierać je do wskazanego katalogu „na boku” (np. jakiś podkatalog w katalogu `C:\Install`), a następnie z podkatalogu `bin` w tak wybranym miejscu spośród ściągniętych przez menedżera plików, odpowiednio będziemy kopiować do katalogu z ręcznie zainstalowanymi programami. Należy tutaj podkreślić, że programy z grupy `MSYS` nie są ściągane do podkatalogu `bin` wskazanej ścieżki „na boku”, jak te z grupy `MinGW`, ale głębiej, do podkatalogu `msys\1.0\bin` i z tego katalogu wybrane pliki należy kopiować do przygotowanego katalogu z programami instalowanymi ręcznie. Niektóre biblioteki mogą się powtarzać i jeśli są już skopiowane, można ich ponownie nie kopiować.

Na programy instalowane ręcznie można wybrać np. `c:\Program Files\Linux\bin` albo wprost `C:\bin`. Ten katalog z binariami powinien być umieszczony na liście ścieżek z programami, wpisanej do zmiennej systemowej `PATH`; robimy to w ustawieniach `System | Zaawansowane ustawienia systemu | Zmienne środowiskowe | Zmienne systemowe | PATH`, dodając ścieżkę (np. wspomnianą `C:\Program Files\Linux\bin`), najlepiej na początku. Należy też zauważyć, że otwarte już sesje wiersza poleceń nie zauważą zmiany zmiennej środowiskowej `PATH` i trzeba je zrestartować.

- aby zainstalować `ls.exe`, należy wybrać w `MSYS | MSYS Base System` pakiet `msys-coreutil-bin`. Po ściągnięciu trzeba zainstalować ręcznie pliki `ls.exe`, `msys-1.0.dll`, `msys-iconv-2.dll` i `msys-intl-8.dll`.
- aby zainstalować `grep.exe`, należy użyć ściągniętego już w `MSYS | MSYS Base System` pakietu `msys-coreutil-bin` i zainstalować zeń ręcznie pliki `grep.exe`, `egrep.exe`, `fgrep.exe`, `msys-1.0.dll` i `msys-intl-8.dll`.
- aby zainstalować jakiś inny program z tych środowisk, należy wyszukać go w pakietach, pakiet ściągnąć i jego programy skopiować do katalogu instalacji ręcznych; jeśli uruchamiając go otrzymamy komunikat o brakującej bibliotece, należy ją odnaleźć, podkopiować i spróbować program uruchomić jeszcze raz. Np. `wget.exe` jest w `msys-wget-bin` ale przy pierwszym uruchomieniu poskarży się na brak biblioteki `msys-ssl-1.0.0.dll`, potem zawoła o `msys-crypto-1.0.0.dll`. Do niektórych skryptów opcjonalnych (tych dla wersji personalnych czy mailowania) potrzebne jest `cat.exe` oraz `rm.exe` z bibliotekami `libiconv2.dll` oraz `libintl3.dll`.

- b) **AWK** służy do przetwarzania plików tekstowych, ich filtrowania i automatycznego korygowania. Ja używam także skryptów AWKowych do sortowania indeksów.

Ścieżka <https://sourceforge.net/p/ezwinports/activity/> zawiera najnowsze porty narzędzi linuksowych pod Windows; szukamy na niej Gnu AWKa. Można też go znaleźć na liście plików do pobrania <https://sourceforge.net/projects/ezwinports/files/>. Aktualnie najnowszą wersję znalazłem na ścieżce <https://sourceforge.net/projects/ezwinports/files/gawk-5.1.1-w32-bin.zip/download>.

Szukanie nowej wersji jest istotne o tyle, że AWKa używamy do dość specyficznego sortowania indeksów alfabetycznych i analitycznych, uwzględniających ignorowanie w sortowanych ciągach interpunkcji oraz do wyłuskiwania ciągów liczbowych i określania porządku według wartości liczb. Do tego potrzebna jest funkcja wbudowana `asort` o nieprzestarzałej specyfikacji ISO z możliwością zdefiniowania własnej funkcji do porównywania kluczy sortowania. Obecna wersja AWKa dostępna w tym projekcie ma numer wersji 5.1.1 ale nasze skrypty powinny działać dobrze również na wersji AWKa 4.2.1.

AWKa należy zainstalować ręcznie w katalogu z programami (np. we wspomnianym wyżej `c:\Program Files\Linux\bin`), ale tym razem poprzez wypakowanie do takiego katalogu zawartości katalogu `bin` z zipa z dystrybucją AWKa (np. `gawk-5.1.1-w32-bin.zip`).

To nie wystarczy, bowiem ten AWK potrzebuje biblioteki GCC `libgcc_s_dw2-1.dll`, ale autor projektu `ezwinports` nie uznał się za uprawnionego do redystrybucji bibliotek kompilatora GCC i tę bibliotekę zalecił podkopiować z innego miejsca. Dlatego bierzemy ją z pakietu `MinGW` po wybraniu i ściągnięciu pakietu `mingw32-base-bin`. Z podkatalogu `bin` kopiujemy brakującą bibliotekę `libgcc_s_dw2-1.dll` do katalogu z binariami AWKa.

6. Do generacji indeksów słów używamy pakietu `texindy` a ten (jak i kilka innych), chociaż obecny w zainstalowanym `MiKTeXu`, potrzebuje silnika interpretera `perl.exe`. Dlatego potrzebujemy mieć zainstalowany Perl a w systemie Windows domyślnie go nie ma. Instalator pobieramy ze strony <https://www.perl.org/get.html> dla Windows w wersji Truskawka (obecnie `strawberry-perl-5.32.1.1-64bit.msi`) i uruchamiamy go. `Texindy` (i tym samym Perl) jest wywoływany tylko podczas kompilacji śpiewników tekstowych pomiędzy przebiegami `LuaTeXa`.
7. Do składu nut potrzebujemy trzech preprocesorów, kolejno `M-Tx`, `PMX` oraz `MusixTeX`. Preprocesory są wywoływane przed rozpoczęciem `TeXowania`. Są to programy wykonywalne, które mają w niektórych wersjach swoje słabości. Wszystkie te preprocesory powinny być zainstalowane w katalogach wykonywalnych zadeklarowanych na ścieżkach w zmiennej środowiskowej `PATH`, co opisałem wyżej. Programy te są także obecne w dystrybucji `MiKTeXa`, którego katalogi z binariami są wpisane do ścieżek `PATH` przez instalator `MiKTeXa`. Niestety, ze względu na braki w panowaniu nad rozwojem tych narzędzi (każde ma innego autora a zbiór użytkowników jest stosunkowo niewielki), nie każda wersja radzi sobie z materiałem wejściowym.
 - a) pakiet `M-Tx` służy do podpisywania tekstu pod nutami. Notacja, którą się posługuje jest rodem z pakietu `PMX`, czyli nieco uproszczona w stosunku do notacji `TeXowej` natywnego dla `TeXa` i `LaTeXa` pakietu `MusixTeX`. Aby włożyć tekst podpisany pod nutami do kodu w formacie `PMX`, pakiet `MT-x` potrzebuje preprocesora (zewnętrznego binarnego programu przetwarzającego) `prepmx.exe`, obecnego w `MiKTeXu` na ścieżce `C:\Program Files\MiKTeX 2.9\miktex\bin\x64\prepmx.exe` w najnowszej wersji 0.63. Z kilku powodów *ta wersja jest nieprzydatna*. Najsamopierw dlatego, że poczynając od wersji 0.61 autor przepisał od nowa kod preprocesora, wprowadzając ograniczenia, których przedtem nie było a które swoimi wymogami zabraniają pewnych funkcjonalności, których używamy (np. nie pozwala nad pauzami wielotaktowymi umieszczać tekstu, w którym my umieszczamy akordy, które się czasami pojawiają właśnie nad takimi pauzami). Drugim powodem jest poważna niestabilność nowszych wersji, które zawierają różne błędy i zwyczajnie się wałą albo zawieszają. W szczególności w rekomendowanej instalacji `MiKTeXa` z 2018 roku wersja 64-bitowa tego programu wiesza się na twardo i niczego nie przetwarza ani nie produkuje na wyjściu. Wersja aktualna już się nie wiesza, ale wywala się w wielu miejscach z komunikatem, że jest to błąd wewnętrzny. Dość, że z jakichś powodów ta wersja nie chce przetwarzać naszych nut i trzeba by było metodą prób i błędów omijać problemy. Rozwiązaniem szybszym i skuteczniejszym jest zainstalowanie wersji stabilnej `prepmx.exe` 0.60d w jakimś oddzielnym katalogu z binariami (ja to instaluję w katalogu `c:\bin`) i ustawić ten katalog na ścieżkach `PATH` na pierwszej pozycji, przed innymi (także przed ścieżką binariów `MiKTeXa`). Wprawdzie jest to kod 32-bitowy ale działa dobrze, natomiast preprocessing nie stanowi istotnej części kompilacji i prawie nie wpływa na czas budowy ani całości ani samych nut. Wersję stabilną preprocesora `prepmx.exe` znajdziemy na portalu WIMA: Werner Icking Music Archive w katalogu <https://www.icking-music-archive.org/software/mtx/>. Zajmują się tam ogólnie nutami oraz wszelkimi narzędziami związanymi z notacją muzyczną. W powyższym katalogu z `MT-x` należy znaleźć i pobrać binaria pod win32 dla wersji 0.60d, czyli plik <https://www.icking-music-archive.org/software/mtx/mtxP060b-win32.zip>. W tym ZIP-ie znajdziemy plik `prepmx.exe`, którym należy zasłonić na ścieżkach `PATH` ten z dystrybucji `MiKTeXa`.
 - b) pakiet `PMX` umożliwia prostszy zapis składu nut, niż ten proponowany przez `MusixTeX`. Jest też wykorzystywany przez `MT-x`, który podkłada tekst generując jednocześnie kod `PMX`. On także używa do przetworzenia na kod `MusixTeXa` preprocesora, tym razem o nazwie `pmxab.exe`. Tutaj także autor zmieniał różne rzeczy i trudno orzec, dlaczego w niektórych sytuacjach preprocesor obecny w dystrybucji `MiKTeXa` sobie nie radzi albo i się wywraca. Być może szczegółowa analiza wykorzystywanych funkcjonalności pozwoli-

łaby obejść niedoskonałości najnowszych wersji tegoż preprocesora. Jednak przy ośmiuset stronach gotowych nut nie ma powodu, aby ponownie rewidować i zmieniać kod źródłowy. Dlatego stosuję chwyt podobny, jak wyżej, czyli używam ostatniej znanej i sprawdzonej wersji stabilnej `pmxab.exe` 2.70, przesłaniając nim wersję obecną w binariach MiKTeXa. Znajdziemy go w <https://www.icking-music-archive.org/software/pmx/> pod nazwą `pmx270.zip` czyli na ścieżce <https://www.icking-music-archive.org/software/pmx/pmx270.zip>. Z tego ZIPa należy wypakować plik `pmxab.exe` i umieścić w katalogu przesłaniającym na ścieżkach `PATH` binaria z MiKTeXa (np. wspomnianym `C:\bin`).

- c) Należy podkreślić, że MusixTeX wydaje się być stabilny i jego preprocesor, a właściwie bardziej rekalkulator odstępów `musixflx.exe`, wydaje się działać prawidłowo i nie potrzebuje żadnych sztuczek ani przesłaniania innymi wersjami. Podobno był jakiś projekt aby w wypadku LuaTeXa rozmieszczanie nut na bieżąco rozliczał jakiś kod w Lua, ale nie szukałem wystarczająco cierpliwie takiego rozwiązania. Rzeczywiście, wynikiem działań preprocesorów jest kod TeXowy, który nie potrzebuje dalszego przetworzenia, ale aby dawał prawidłowy skład, rekalkulator musi zostać wywołany pomiędzy przebiegami składu i rozmieścić właściwie nuty w taktach i w pięciolinii.

8. Jeżeli będziemy chcieli optymalizować rozmiar generowanych PDFów, będziemy potrzebowali:

- a) płatnego **Adobe Acrobat Pro**; skład śpiewnika w PDFie ma około 10 MB a po optymalizacji ma ponad trzykrotnie mniej a nuty także są niemal dwukrotnie mniejsze po zoptymalizowaniu; ze wszystkich narzędzi optymalizujących pliki PDF tylko Adobe Acrobat radzi sobie z zadaniem rzeczywiście dobrze
- b) aby automatyzować optymalizację PDFów, niezbędny będzie **Autohotkey**, który potrafi sterować programami okienkowymi i steruje też skutecznie Acrobatem podczas optymalizacji. Pakiet możemy pobrać ze strony <https://www.autohotkey.com>. W naszych źródłach jest skrypt `autohotkey.ahk`, który steruje Acrobatem, za pośrednictwem tego pakietu, jednak stałe oczekiwania w tym skrypcie należy dopasować do mocy obliczeniowej każdego komputera.
- c) Jeżeli skrypty składające mają uruchamiać automatyczną optymalizację, w zmiennych środowiskowych systemu (tam, gdzie wspomniana przy ręcznej instalacji binariów, zmienna `PATH`) należy dodać nową zmienną `optimisepdf` i nadać jej wartość 1. Należy pamiętać, że proces budowy będzie trwał istotnie dłużej. Wielordzeniowe procesory budowę (skład) wielu wersji śpiewników mogą robić równolegle. Acrobat może optymalizować naraz tylko jeden plik PDF. Co gorsza, automatyczne sterowanie Acrobatem zarządza fokusem okienka i nie można w tym samym czasie pracować na kontekście graficznym, natomiast funkcjonalność optymalizacji wsadowej zazwyczaj się wywraca. Przy częstych budowach warto rozważyć postawienie maszyny wirtualnej i na niej startować budowy, które chociaż będą trwały zapewne nieco dłużej, to nie będą blokować nam całego kontekstu graficznego komputera.
- d) Acrobat po instalacji lubi zadawać pytania o dodatkowe funkcjonalności, jak np. odczyt treści, przed ich pierwszym użyciem. Aby automatyczna optymalizacja nie była nimi zakłócana, warto wystartować raz ręcznie skrypt `autohotkey.ahk` z jakimkolwiek wskazanym w parametrze plikiem PDF i odpowiadając zaznaczyć: nie pytaj o to więcej.
- e) Podczas optymalizacji lepiej nie psuć rozdzielczości ikony na stronie tytułowej. Dlatego skrypt `autohotkey.ahk` w okienku **Optymalizator PDF** emituje klawisz „O”, aby wybrać ustawienia nie zmniejszające rozdzielczości obrazów kolorowych. Warto przygotować wcześniej taki zbiór ustawień z nazwą zaczynającą się od litery „O”, np. **OptimiseSongbook**, wychodząc od ustawień standardowych, wyłączając zmniejszanie rozdzielczości obrazów kolorowych i zapisując ustawienia pod wspomnianą nazwą.

9. Pakiet PuTTY potrzebny jest do aktualizacji przebudowanych plików na serwerze internetowym. Poszukiwanie pakietu możemy rozpocząć od strony <https://www.putty.org/>, która nas odeśle na stronę twórcy <https://www.chiark.greenend.org.uk/~sgtatham/putty/latest>.

html. Tamże znajdziemy instalator ostatniej wersji wszystkich programów zestawiających połączenie tunelowe SSH. Jest to jedna z metod aktualizacji plików w repozytorium strony www (obecnie <http://spiewnik.andrzej.odyniec.info>). Aby skrypty zadziałały, należy sprawdzić, czy instalator dodał ścieżkę, pod którą programy z pakietu PuTTY się zainstalowały (zazwyczaj `c:\Program Files\PuTTY\`) do zmiennej środowiskowej PATH, gdyż z tym bywało różnie.

Aby aktualizacja plików na serwerze działała, należy zadbać o wygenerowanie i umieszczenie kluczy prywatnych w podkatalogu `keys` katalogu instalacyjnego PuTTY, o wyeksportowanie klucza publicznego oraz o to, aby klucz publiczny był zainstalowany na serwerze, zazwyczaj w pliku `.ssh/authorized_keys`. Następnie należy zbudować konfigurację pod jakąś nazwą, wpisując w niej nazwę loginu (w `Connection | Data | Auto-login username`) i odniesienie do pliku klucza prywatnego (w `Connection | SSH | Auth | Private key file for authentication`). W wywołaniach programów z dystrybucji PuTTY należy wskazać w parametrze `-load` nazwę konfiguracji.

Konfiguracja jest zapamiętywana przez PuTTY w rejestrach systemu Windows pod kluczem `HKEY_CURRENT_USER\SOFTWARE\SimonTatham\PuTTY\Sessions` i w wypadku przenoszenia się na inny komputer, można klucze konfiguracji wyeksportować i zaimportować w nowym miejscu, byle ścieżka do klucza w polu konfiguracji `PublicKeyFile` się zgadzała.

Wszelkie formy loginu na zdalnych serwerach przez samo hasło (bez osłony kryptograficznej uwierzytelnianej kluczem prywatnym) są bardzo niebezpieczne.

10. Za synchronizację przy pomocy pakietu PuTTY z serwerem odpowiadają skrypty wywoływane jawnie lub niejawnie przez skrypty budowy. Nie wszystkie skrypty związane z synchronizacją i budową są dystrybuowane ze źródłami. Należy mieć świadomość, że skrypty aktualizujące są zależne od architektury serwera i od skryptów wspomagających na serwerze. Moje skrypty współpracują dobrze z moim serwerem wynajętym w `home.pl`, chociaż swojego czasu współpracowały dobrze z innym serwerem linuksowym.

- a) Pierwszym takim plikiem, (zawartym w źródłach) jest `frontcomm-original.sftp`. Jest to prototyp skryptu z poleceniami dla programu z pakietu PuTTY o nazwie `psftp.exe`, który jest klientem bezpiecznego protokołu transferu plików Secure FTP. Zawiera polecenie przejścia do katalogu śpiewnika (komenda `cd`) a potem wysłania tam zbudowanych plików (komendami `put`). Obecnie ten plik ma następującą zawartość:

```
cd public_html/guru/sp
put SP.pdf
put SP-layers.pdf
put SP-alpha.pdf
put SP-layers-alpha.pdf
put SPnochords.pdf
put SPnochords-alpha.pdf
put SPpocket.pdf
put SPpocket-alpha.pdf
put SPnarrow.pdf
put SPnarrow-alpha.pdf
put SPkikosq.pdf
put SPkikosq-layers.pdf
put SPkikosq-alpha.pdf
put SPkikosq-layers-alpha.pdf
put SPtab34.pdf
put SPtab34-alpha.pdf
put SP-mono.pdf
put SP-mono-alpha.pdf
put broszury.pdf
put broszury-alpha.pdf
put broszurynochords.pdf
put broszurynochords-alpha.pdf
put broszurypocket.pdf
put broszurypocket-alpha.pdf
put broszurnarrow.pdf
put broszurnarrow-alpha.pdf
put liturg.pdf
put liturg-alpha.pdf
put spiewnik.pdf
put spiewnik-alpha.pdf
```

```

put spiewnik-layers-alpha.pdf
put SpiewnikNeokatechumenalnySrc.zip
put NeokatechumenalneNutyLiturgiczneSrc.zip
put NutySpiewowNeokatechumenalnychSrc.zip
put NeokatechumenalneNutyLiturgiczneMIDI.zip
put NutySpiewowNeokatechumenalnychMIDI.zip

```

Ten plik nie zawiera komend `put` dla wersji osobistych. Zostanie on wczytany na początku budowy przez skrypt `mpbuni.awk` i po uwzględnieniu listy wersji osobistych utworzonych z pliku `tps.tex` oraz ewentualnie tego, czy budowane są tylko wersje `alpha` lub tylko nuty, zostanie z niektórych komend `put` przetrzebiony a innymi uzupełniony i umieszczony pod nazwą `frontcomm.sftp` (podczas normalnej budowy), `frontcommn.sftp` (podczas budowy samych nut) lub `frontcommns.sftp` (podczas budowy tylko nut śpiewnikowych).

- b) Lista wersji osobistych jest w pliku `tps.tex` i ten plik **nie jest załączany w paczce ze źródłami**, gdyż mendi, którym wydaje się, że mają prawo do rządów dusz, wykorzystują informacje o osobach aby gnoić tych, którzy zamówili sobie wersje osobiste. Z tego samego powodu wersje personalne śpiewnika są ukryte pod indeksem strony, aby wścibski władczy katechista nie mógł podejrzec, kto sobie zamawiał wersję personalną i nie mógł potem takiego kantora prześladować. Jeśli z jakichś powodów ktoś będzie potrzebował przebudować zamówione do tej pory wersje osobiste, musi zdobyć istniejący plik `tps.tex` albo ode mnie albo z katalogu śpiewnika na serwerze, gdzie jest on wykorzystywany i korygowany przez skrypt `transpozycje.html`, chociaż jest ukryty pod indeksem (jest indeksem przesłonięty).
- c) Synchronizację uruchamiają następujące pliki wsadowe (w `CMD`, nie w `Powershellu`) o nazwach: `syncsftp.bat`, `syncsftpn.bat` i `syncsftpns.bat` odpowiednio po budowie całości, samych nut i tylko nut śpiewnikowych.
- d) Na potrzeby budowy i synchronizacji właśnie zamówionej lub zmodyfikowanej wersji personalnej (po otrzymaniu powiadomienia pocztą z serwera) przewidzieliśmy skrypt (tym razem w `CMD` a nie w `Powershellu`) przed rozpoczęciem budowy: `syncsftpssendpre.bat` oraz po jej zakończeniu: `syncsftpssendpost.bat`. Skrypty te posługują się skryptami komend powłoki serwera dla `putty.exe` i komend `ftp` dla `psftp.exe`, o tych samych nazwach bazowych z rozszerzeniami odpowiednio `.ssh` oraz `.sftp`.
- e) Na potrzeby wysyłki powiadomień pocztą elektroniczną mamy skrypty `sending.ps1` oraz `sendindg.awk` biorące szablon treści maila z pliku `send-message.txt`. W końcu użyją one skryptu `send.ps1`, który nie jest dołączany do źródeł, gdyż w jego treści jest uwierzytelnienie w serwerze nadawczym poczty. Treść tego skryptu należy dopasować do parametrów poczty nadawcy. Treść skryptu (z pominięciem uwierzytelnień) jest następująca:

```

$id=$args[0]
$mail=$args[1]
$name=$args[2]
$rules=$args[3]
awk -v id="$id" -v name="$name" -v rules="$rules" -v file="send-message.txy" -f send.awk send-message.txt
awk -v id="$id" -v name="$name" -v rules="$rules" -v file="send-subject.txy" -f send.awk send-subject.txt
$Subject = Get-Content .\send-subject.txy -Encoding utf8
$Body = Get-Content .\send-message.txy -Encoding utf8 -Raw
$EmailFrom = "Andrzej Odyniec" <andrzej@odyniec.info>
$EmailTo = ''+$name+' ' <'+$mail+'>'
#$EmailTo = ''+$name+' ' <andrzej.odyniec@gmail.com>
$EmailBcc = "Andrzej Odyniec" <andrzej@odyniec.info>
$SMTPServer = "serwer1762249.home.pl"
$SMTPClient = New-Object Net.Mail.SmtpClient($SMTPServer, 587)
$SMTPClient.EnableSsl = $true
$SMTPClient.Credentials = New-Object System.Net.NetworkCredential("username", "password");

$secpasswd = ConvertTo-SecureString "password" -AsPlainText -Force
$cred = New-Object System.Management.Automation.PSCredential ("username", $secpasswd)
Send-MailMessage -From $EmailFrom -To $EmailTo -Bcc $EmailBcc -Encoding utf8 -Subject $Subject
-Body $Body -SmtpServer $SMTPServer -Port 587 -UseSsl -Credential $cred
-DeliveryNotificationOption 'OnSuccess','OnFailure','Delay'

$datesent = get-date -Format "yyyyMMdd-hhmmss"
echo "$datesent+$id" | Out-File -FilePath sent.txt -Append -Encoding ascii
del send-message.txy
del send-subject.txy

```

11. W dystrybucji mogą być niedostępne wszystkie skrypty budowy śpiewników i nut albo pliki wspomagające tę budowę.

Skrypty budowy przyjmują, że materiały źródłowe znajdują się w katalogu `spiewnik1250work` (nazwa historyczna) natomiast budowa realizowana jest w katalogu `..\spiewnikUTF8` (leżącym obok). Pełna budowa opróżnia ten katalog, pakuje do paczki ZIP ze źródłami wszystkie niezbędne pliki z katalogu źródłowego i rozpakowuje go w katalogu budowy. Tam realizowana jest budowa w oparciu wyłącznie o pliki z paczki ze źródłami. Jednak niektóre pliki, zawierające uwierzytelnienia do serwerów czy dane osób, nie mogą być dystrybuowane ze źródłami. Nie są one uruchamiane z katalogu budowy ale z katalogu z materiałami źródłowymi i tam muszą być umieszczone. Same skrypty budowy wołają skrypty sterujące preprocessingiem, kompilacją wieloprzebiegową oraz optymalizacją a do synchronizacji używają Mutex-ów.

Skrypty budowy startujemy z okienka Powershella z katalogu ze źródłami. Ja takie okienko otwieram komendą:

```
start powershell -ExecutionPolicy Unrestricted -NoExit  
-Command "cd C:\Users\guru\Documents\Community\spiewnik1250work"
```

- a) do uruchamiania budowy służy skrypt `buildtimeuni.ps1`, który ma trzy argumenty:
- pierwszy argument specyfikuje ograniczenia budowy dla wersji osobistych; łańcuch tekstowy tego argumentu:
 - może na początku zawierać liczbę, która ogranicza budowę do tych kantorów, którzy w pliku `tps.tex` mają komendę `\PersonName` zaczynającą się od kolumny o tym numerze; podanie liczby 0 oznacza rezygnację z budowy wersji personalnych, natomiast podanie tekstu `all` oznacza budowanie wszystkich wersji personalnych
 - może zawierać słowo `alpha`, co ogranicza wersje personalne tylko do tych kantorów, którzy wyspecyfikowali w swojej wersji opcję `alpha` a więc chcą śpiewnika w układzie alfabetycznym
 - może zawierać słowo `layers`, co oznacza budowę wersji osobistych tylko dla tych kantorów, którzy ustawili dla siebie opcję `layers`, a więc chcą wersji osobistej z wieloma tonacjami na warstwach;
 - drugi argument specyfikuje ograniczenia budowy wersji ogólnych i może zawierać słowo `alpha` albo może być pusty; jeśli zawiera słowo `alpha`, będą budowane wersje ogólne tylko dla nowego układu alfabetycznego;
 - trzeci argument może być pusty; jeśli jednak ma wartość 1, upload rezultatów budowy na serwer wykonywany jest automatycznie, bez pytania operatora; oczywiście, jeśli drugi argument musiałby być pustym przy trzecim o wartości 1, to należy podać w miejscu drugiego dwa apostrofy, gdyż argumenty w Powershellu są pozycyjne.

Należy liczyć się z tym, że budowa potrwa od 10 minut (dla samych nut) i 25 minut (dla wersji ogólnych) na szybkich maszynach aż do 2,5 godziny dla wszystkich wersji personalnych na wolniejszych maszynach.

Kompletna budowa z optymalizacją wszystkich wersji ogólnych plus trzydziestu dwóch wersji personalnych, niemal wszystko z układami broszurowymi, na ośmiokontekstowej maszynie 4+ GHz (cztery rdzenie plus hyperthreading) kosztowała ~85 minut. Budowa w takich samych warunkach ale bez optymalizacji plików PDF kosztowała ponad dwukrotnie mniej, bo ~38 minut, natomiast na osiemnastu rdzeniach z hyperthreadingiem (czyli w sumie na trzydziestu sześciu kontekstach) sama budowa bez optymalizacji zakończyła się po 16 minutach. Jednak zbudowane PDFy bez optymalizacji ważą około 650 MB, gdy tymczasem zoptymalizowane tylko 260 MB.

Maszyny z większą liczbą rdzeni są w stanie skrócić samo T_EXowanie ale optymalizacja i tak, ze względu na ograniczenia Acrobat, idzie szeregowo i rozłącznie w sekcji krytycznej.

- b) skrypt `buildnotestimeuni.ps1` buduje wyłącznie nuty, przy czym ma układ argumentów identyczny z powyższym; pierwszy argument nie ma znaczenia, gdyż decyduje jedynie

o wersjach osobistych śpiewnika, które tutaj nie będą budowane, jednak musi być podany, jako pusty, jeśli drugi argument ogranicza budowę nut wyłącznie do tych w wersji alfabetycznej.

Ponieważ budowa nut używa wcześniej zbudowanego śpiewnika do załączenia go i późniejszego otwierania przez łączy do śpiewnika i odwrotnie, po budowie śpiewników wyłącznie dla wersji `alpha`, nuty także muszą być budowane wyłącznie dla wersji `alpha`.

- c) skrypt `buildtimepouni.ps1` służy do dobudowania lub ponownej budowy wyłącznie wersji personalnych dla kantorów, którzy coś zmienili w swojej osobistej wersji, czy w opcjach czy w tonacjach. Dlatego należy pomiędzy zmianami w specyfikacjach na serwerze a rozpoczęciem dobudowy zsynchronizować z katalogiem lokalnym plik `tps.tex` z serwera. Robimy to wcześniej przy pomocy skryptu `syncsftpssendpre.bat`. Przy pomocy skryptu `tpscomp.awk` zostanie porównany zsynchronizowany z serwerem plik `tps.tex` oraz ostatnia jego wersja użyta do budowy, po czym wystartowana zostanie budowa wyłącznie dla tych kantorów, u których coś uległo zmianie.
- d) skrypt `buildpostmsgs.ps1` robi synchronizację z serwerem aby ustalić, których użytkowników należy powiadomić o zmianach, aby pobrali aktualne wersje, następnie wysyła do nich powiadomienia po czym realizuje synchronizację w drugą stronę aby zaktualizować na serwerze informacje o wysyłkach i w przyszłości uniknąć ponownej wysyłki wiadomości.

12. Ta pozycja nie jest obowiązkowa, chociaż może się przydać.

Do tworzenia skompresowanych paczek ze źródłami czy z plikami midi oraz do rozpakowywania źródeł w celu budowy potrzebne są programy wiersza poleceń `zip.exe` oraz `unzip.exe`. Takie programy należą do pakietu Info-ZIP a ich źródła można pobrać z portalu <https://sourceforge.net/projects/infozip/>. Niestety, nie ma tam gotowych kompilatów pod Windows, więc trzeba ich poszukać w sieci. Ja znalazłem je tutaj: <http://stahlworks.com/dev/index.php?tool=zipunzip>, niemniej może się okazać, że będą kiedyś niedostępne. Wtedy należy znaleźć ponownie kogoś, kto potrafi je skompilować. Kompilat `zipa` można znaleźć w binariach MiKTeXa ale `unzipa` tam już nie ma.

Obecnie posługujemy się komendami wbudowanymi w windowsową powłokę Powershell o nazwach `Compress-Archive` oraz `Expand-Archive`. Komendy Powershella mają nieco inną składnię — nazwy plików do kompresji muszą wystąpić w parametrze `-Path` i być oddzielonymi przecinkami lub podane jako tablica (która może być wypełniona z oddzielaniem pozycji znakiem nowej linii zamiast przecinka), natomiast nazwa tworzonego archiwum występuje w parametrze `-DestinationPath`. Przy dekompresji `-Path` podaje nazwę archiwum a `-DestinationPath` nazwę katalogu docelowego (inaczej będzie to katalog o nazwie archiwum). Na dodatek te wbudowane applety, w przeciwieństwie do `zip.exe`, zatrzymują się, jeśli któregoś z listy plików do spakowania zabraknie (może to i dobrze).

13. Drogi czytelniku, jeśli przeraziły cię powyższe instrukcje, jak zdobyć, pobrać i ręcznie zainstalować kilka powyżej opisanych pakietów i programów, i zamiast instalować rzeczy krok po kroku, doczytałeś do tego miejsca, to możesz rozważyć skorzystanie z moich zbiorów. Zebrane komponenty można znaleźć w podkatalogu `source-materials` w katalogu śpiewnika pod nazwą `Environment.zip`. Plik ten waży około 2,7GB i zawiera także (a może przede wszystkim) instalację MiKTeXa z roku 2018, o której było na początku. W razie wątpliwości, czy nie podkładał tam jakiegoś pegasusa, zawsze możesz skonfrontować moje zbiory ze wskazanymi powyżej witrynami.

Innym rozwiązaniem jest posłużenie się gotową maszyną wirtualną przystosowaną do budowy. Jednak rozmiar tej maszyny przekracza 60 GB i tym samym po nią należy się zgłosić do mnie.