Projekt SCZR

Projekt systemu realizującego modulacje dźwięku

Zespół: Jakub Gałat, Jakub Gruszecki, Konrad Kulesza

Opis systemu:

System będzie zrealizowany na komputerze, pracującym pod kontrolą systemu Linux. Ma on na celu zapewnić dostarczenie do użytkownika zmodulowanego dźwięku, którego źródłem będzie urządzenie wejściowe użytkownika. Procesy realizujące system kontaktują się ze sobą w czasie rzeczywistym, celem zapewnienia płynności odbieranego przez użytkownika dźwięku. W celu zapobiegania sytuacjom, w których mogłoby dojść do zawieszenia systemu, nasz system dostanie wydzieloną ograniczoną liczbę rdzeni.

Procesy:

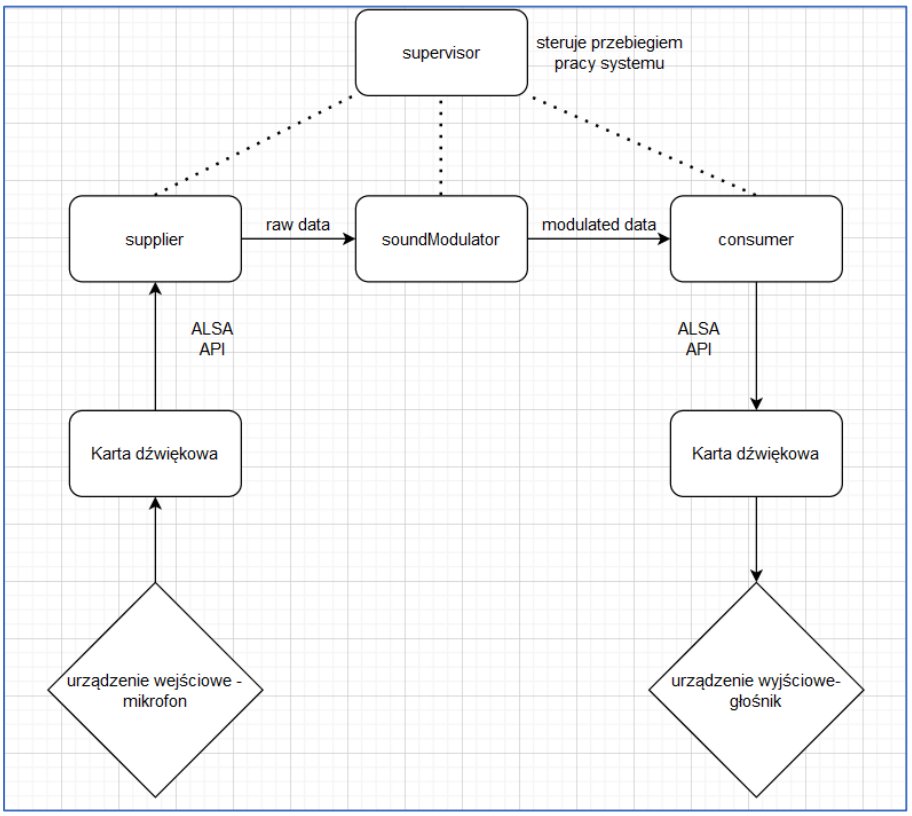
supplier – producent danych. Odbiera dane z karty dźwiękowej w pętli za pomocą funkcji biblioteki ALSA i dostarcza je do procesu przetwarzającego. Na tą chwilę używamy mechanizmu pamięci współdzielonej, w której supplier zapisuje dane pobrane z pomocą funkcji bibliotecznej z urządzenia wejściowego.

soundModulator – proces przetwarzający dane. Przetwarza dane w sposób narzucony przez supervisora, przekazuje dane do consumera. Obecnie stworzyliśmy prostą funkcje przetwarzającą dane, której celem jest w dużej mierze testowanie systemu. W następnym kroku zdecydowaliśmy się skorzystać z biblioteki Rubber Band Library:

https://breakfastquay.com/rubberband/code-doc/classRubberBand_1_1RubberBandStretcher.html

consumer – proces przekazujący dane do użytkownika. Przekazuje dane na wyjście karty dźwiękowej za pomocą funkcji biblioteki ALSA. Na ten moment proces ten korzysta z mechanizmu pamięci współdzielonej, w której znajdują się dane przekazane przez suppliera. W kolejnym kroku pomiędzy supplierem a consumerem, planujemy wykorzystać proces soundModulator do modyfikiacji danych. Wykorzystamy dwa fragmenty pamięci współdzielonej (dla komunikacji supplier-soundModulator i soundModulator-consumer) oraz kolejkę komunikatów powiadamiającą o dostępności danych do dalszego przetworzenia.

supervisor – proces odpowiedzialny za interakcję z użytkownikiem. Zarządza działaniem całego systemu. Jest to program, który jest wywoływany przez użytkownika, udostępnia więc też odpowiedni interfejs do modyfikacji parametrów algorytmu przetwarzania dźwięku. W przypadku testowania kolejki komunikatów (której mechanizm również rozważamy przetestować) będzie on inicjował kolejki i przekazywał identyfikatory do odpowiednich procesów. Proces odpowiada również za zabezpieczenie systemu w przypadku awarii jednego z procesów potomnych i restartuje system w razie potrzeby.



Pomiary:

Celem pomiarów jest zmierzenie latencji pomiędzy wyjściem a wejściem karty dźwiękowej oraz czasu poszczególnych etapów przetwarzania dźwięku (przesył danych pomiędzy procesami, obróbka dźwięku) przy zastosowaniu mechanizmu pamięci współdzielonej.

Pomiary będą uwzględniać dwa przypadki synchronizacji: kolejki komunikatów, semaforów.

Dodatkowo sprawdzimy skuteczność naszego systemu dla dwóch rodzajów trybu szeregowania: ***SCHED_FIFO*** i ***SCHED_RR***.

Pomiary zostaną wykonane przy obciążeniu procesora przy użyciu programu GenLoad oraz bez obciążenia.

Czas trwania etapu zostanie zmierzony poprzez ***std::chrono::high_resolution_clock::now().***

Wyniki pomiarów zostaną zaprezentowane w postaci tabeli i wykresów. Do dokumentacji zostaną również dołączone wnioski i spostrzeżenia wyciągnięte z zaobserwowanych wyników.

Na tą chwilę udało nam się wykonać pomiar latencji pomiędzy wejściem a wyjściem dźwięku, z pomocą specjalnego programu korzystającego z biblioteki ALSA.

Aktualny stan pracy:

Udało nam się zapisać dźwięk z urządzenia wejściowego do pliku i odtworzyć go za pomocą urządzenia wyjściowego za pomocą biblioteki ALSA.

Podzieliliśmy procesy na rdzenie procesora tak, że supplier i supervisor są na 1 rdzeniu, a consumer i voiceModulator na 2 rdzeniu. Dzięki temu w razie dużego obciążenia tych rdzeni nie utracimy kontroli nad systemem operacyjnym.

Trzonem naszych starań było poświęcenie czasu na poszukiwanie odpowiednich bibliotek, które umożliwią podzielenie operacji czytania danych z urządzenia wejściowego i wypisywania na urządzenie wyjściowe na dwie oddzielne funkcje, tak żeby moc skorzystać z odpowiednich funkcji w odpowiadających procesach.

Aktualny stan pracy znajduje się na repozytorium:
https://github.com/Demongo2009/SCZR_VoiceChangerSystem