VYUŽITÍ WEBASSEMBLY PRO WEBOVÉ APLIKACE

BEDŘICH SCHINDLER

CÍL

- Seznámení s WebAssembly
- Popis problémů, které WebAssembly řeší a jeho alternativy
- Demonstrace kroků nezbytných ke kompilace kódu v C++ do WebAssembly a demonstrace použití na straně klientského JavaScriptu ve webové aplikaci
- Ukázka daných postupů na dvou existujících knihovnách

WEBASSEMBLY

- Nízkoúrovňový jazyk s kompaktním binárním formátem navržený pro spuštění v moderních webových prohlížečích
- Umožňuje provádění vysoce výkonného kódu ve webovém prohlížeči rychlostem podobným nativnímu strojovému kódu
- Existuje jak textový formát (WAT), tak binární formát (WASM)
- Je možné využít jako cíl kompilace z jazyků jako jsou C/C++, Python, Go atd.
- Alternativou asm.js

KOMPILACE DO WASM

- Existují kompilátory pro téměř všechny jazyky, některé jsou oficiálně podporované, některé nikoliv (často mají jen částečnou implementaci jazyka)
- V rámci práce byl využíván kompilátor Emscripten (em++), který umožňuje kompilovat kód z C a C++ do WASM

POUŽITÍ VE WEBOVÉM PROSTŘEDÍ

- WebAssembly modul (WASM) stáhneme přes funkci *fetch*
- Pomocí WebAssembly.instantiate či WebAssembly.instantiateStreaming vytvoříme instanci modulu, na kterém jsou dostupné exportované funkce
- Emscripten nám dovoluje vygenerovat JS soubor, který zajišťuje výše uvedenou funkcionalitu a přidává množství dalších pomocných funkcí a jeho forma se řídí parametry kompilátoru em++

UKÁZKY POUŽITÍ

- Práce obsahuje základní exportování funkcí přes EMSCRIPTEN_KEEPALIVE, pokročilejší příklady využívají exportování přes EMSCRIPTEN_BINDINGS
- Implementace na webové straně se liší tím, že pokročilejší ukázka používá pro paralelizaci Web Worker API a neblokuje tak hlavní vlákno prohlížeče
- Ukázková aplikace obsahuje implementaci tří C++ knihoven, které jsou mapovány přes EMSCRIPTEN_BINDINGS a jejich použití ve webovém prostředí je demonstrováno na třech primitivním ukázkových aplikacích