Разработка механизма дедупликации страниц памяти JOS

Существенным недостатком текущей реализации JOS является отсутствие механизма дедупликации страниц.

Дедупликация (от лат. deduplicatio — устранение дубликатов) — специализированный метод сжатия массива данных, использующий в качестве алгоритма сжатия исключение дублирующих копий повторяющихся данных.

При создании в программе больших структур данных, занимающих десятки и даже сотни мегабайт (то есть десятки и сотни четырёхмегабайтных страниц памяти), программист зачастую инициализирует весь объём памяти, занятый этими структурами однородным набором данных (например, нулями) и большая часть памяти подолгу ждёт своего использования, занимая при этом дорогую физическую память компьютера.

Другим примером может являться считывание разными процессами одних и тех же файлов в память для быстрого доступа на чтение. В этом случае мы получаем в физической памяти страницы с одинаковым содержанием, принадлежащие разным процессам.

Решить эту проблему может механизм дедупликации страниц памяти. В рамках этого механизма в физической памяти находится лишь одна страница вместо нескольких одинаковых, и все виртуальные адреса одного или нескольких процессов транслируются в физический адрес этой страницы.

При попытке записи на такую страницу создаётся её копия, и уже в неё производится запись.

Основные пункты по реализации данного механизма в JOS:

- Дедупликация памяти проводится специальным процессом, которому разрешён доступ к каталогам страниц всех процессов (проверка прав в envid2env всегда проходит успешно);
- Процессу устанавливается флаг маскирования прерываний;
- Управление дедупликатору передаётся в порядке, определяемом механизмом Round Robin, при этом если с последнего вызова прошло мало времени, и повторное выполнение дедупликации пока не обосновано, процесс немедленно возвращает управление планировщику;

1) Во время дедупликации процесс создаёт хэш-таблицу с разрешением коллизий методом цепочек.
2) Для всех объектов массива envs с полем env_status отличным от ENV_FREE просматриваются все ненулевые записи таблиц страниц в диапазоне [0..UXSTACKTOP]:
   a) При первом проходе в хэш-таблицу в единственном экземпляре добавляются страницы, содержащие флаги PTE_P и PTE_COW;
   b) При втором проходе в хэш-таблицу добавляются страницы, содержащие флаг PTE_P, но не содержащие флаг PTE_COW. При конфликте добавляемая страница побитово сверяется со всеми, присутствующими в цепочке конфликтов, и при совпадении производится процедура, аналогичная написанной в реализованной библиотечной функции fork().

Для тестирования механизма дедупликации предлагается использовать следующий тестовый набор:

1) Запуск наряду с дедупликатором двух процессов, выделяющих и заполняющих нулями по одной странице памяти, проверка того, что после выполнения

дедупликации в таблице трансляции обоим виртуальным адресам будет сопоставлен один физический;
2) То же самое с двумя страницами, выделенными одним процессом;
3) Выполнение теста 1), изменение одной из страниц, проверка корректности копирования страницы.
4) Выполнение теста 1), выделение ещё одной обнулённой страницы и повторная дедупликация. После этого все 3 виртуальных адреса должны ссылаться на один физический.

Контроль соответствия физических страниц виртуальным предлагается осуществлять путём запуска отдельного процесса, печатающего таблицу соответствия виртуальных адресов всех процессов физическим, строящуюся путём обхода каталога и таблиц страниц всех процессов, присутствующих в массиве envs.