

Dave Plummer: Programming, Autism, and Old-School Microsoft Stories | Lex Fridman Podcast #479

Lex Fridman · 2025-08-29 · <https://www.youtube.com/watch?v=HsLgZzgpz9Y>



Выпуск представляет собой глубокое погружение в золотую эру разработки Microsoft вместе с Дэйвом Пламмером — инженером, создавшим Windows Task Manager, нативную поддержку ZIP-архивов и порт легендарного Space Cadet Pinball. Разговор охватывает эволюцию операционных систем от MS-DOS до Windows NT, технические детали оптимизации кода в условиях жестких ограничений памяти и специфику низкоуровневого дебаггинга.

Вторая ключевая тема подкаста — нейроотличия в IT-индустрии. Пламмер подробно разбирает механику работы аутичного мозга, концепцию монотропизма, маскировку и сенсорные перегрузки, объясняя, почему люди с расстройствами аутистического спектра (РАС) часто становятся выдающимися программистами и как им эффективно выстраивать коммуникацию в корпоративной среде.

Ранние годы и путь в Microsoft

Первый опыт программирования начинался с ввода английских слов в интерпретатор 01:33: В 11 лет доступ к компьютеру TRS-80 Model 1 с 4 КБ памяти можно было получить только в магазине RadioShack. Из-за отсутствия знаний синтаксиса первые попытки написать код сводились к вводу обычных английских фраз. Позже на Commodore 64 был написан клон игры Galaga на чистом машинном коде (без ассемблера), где добавление новых инструкций в

середину программы требовало ручного прописывания прыжков (jumps) в конец памяти и обратно, что превращало архитектуру в спагетти-код.

Фактчек: Пламмер написал Task Manager (87 КБ, без C runtime, экономия 96 КБ), ZIP-папки и порт Pinball — □: авторство подтверждено всеми. Цифры (87 КБ, 96 КБ, 5000 итераций физики в секунду) известны только со слов самого Пламмера и независимо не проверяемы — принимать как мемуар, не как измерение.

Постепенный отсев из школы приводит к низкоквалифицированному труду 07:25: Решение бросить школу редко принимается одновременно — это серия мелких решений «не пойти на урок сегодня». Результатом стала работа в ночную смену в 7-Eleven. Поворотной точкой стал инцидент в 40-градусный мороз: при замере уровня бензина деревянным щупом в руки впились тысячи заноз, что заставило осознать необходимость возвращения в систему образования в 21 год.

Коммерческий успех shareware-программ открыл двери в Microsoft 11:38: Для оплаты учебы была написана программа HyperCache — N-way ассоциативный кэш файловой системы для компьютеров Amiga. Ядро передачи данных было написано на ассемблере 68000, остальная логика — на C. Программа продавалась по \$20-40 и разошлась тиражом в несколько тысяч копий. Поиск email-адресов сотрудников Microsoft в регистрационных карточках пользователей HyperCache позволил отправить холодное письмо, которое привело к стажировке в команде MS-DOS.

Успех Microsoft строился на концентрации интеллекта и монополии стандарта 14:49: В 90-е годы компания представляла собой самую мощную концентрацию умных людей в индустрии. Билл Гейтс был одержим идеей «компьютер на каждом столе» и нанимал людей умнее себя. Сделка по включению MS-DOS (очень простой ОС, работающей как командная оболочка без многозадачности) в каждый IBM PC сделала их стандартом, который компания успешно монетизировала десятилетиями.

Эпоха MS-DOS и архитектура Windows NT

Ограничение в 640 КБ оперативной памяти требовало экстремальных хаков 18:28: В MS-DOS любой килобайт, занятый системной утилитой, навсегда отбирался у пользователя по всему миру. Для обхода лимита использовался аппаратный трюк с линией A20 на процессорах x86: комбинируя регистры сегмента и смещения, можно было получить адреса выше 1 МБ, открывая доступ к дополнительным 64 КБ памяти (High Memory Area). Туда были перенесены драйверы кэширования SmartDrive и сжатия DoubleSpace.

Фактчек: НМА через линию A20 даёт 64 КБ выше 1 МБ — □: точнее 64 КБ минус 16 байт. В НМА обычно грузилось ядро DOS (`DOS=HIGH`), SmartDrive/DoubleSpace чаще шли в UMB — но именно их ужатию Пламмер и занимался, суть верна.

Портирование Windows 95 на ядро NT вскрыло ужасающее качество

старого кода 24:50: Перенос графического интерфейса Win95 на архитектуру NT требовал конвертации всех 8-битных ANSI-символов в 16-битный Unicode. Построчный анализ чужого кода выявил строки длиной по 200 символов, обилие нецензурной лексики в комментариях и крайне непрофессиональные архитектурные решения, которые пришлось вычищать.

Дэйв Катлер внедрил в Microsoft стандарты промышленной разработки

26:53: Архитектор ядра Windows NT (перешедший из Digital Equipment Corporation) играл для Windows ту же роль, что Линус Торвалдс для Linux. Он лично проверял качество кода, жестко пресекал попытки внедрения неоптимизированных решений в ядро и заставил разработчиков Microsoft перейти от хакерского стиля к строгим инженерным стандартам.

Невыровненный доступ к памяти замедляет работу Windows до сих пор

30:35: При портировании оболочки на RISC-архитектуру (MIPS) возникла проблема сложения путей (ID lists). Если длина строки была нечетной, следующий адрес оказывался невыровненным. На MIPS это вызывало аппаратное исключение, обработка которого делала чтение строки в 100-1000 раз медленнее. Предложение принудительно выравнивать адреса было отвергнуто в ходе внутренних споров из-за близости релиза, и этот неоптимальный код работает в Windows по сей день.

Фактчек: Невыровненный доступ на MIPS замедлял чтение в 100-1000 раз, и этот код живёт в Windows до сих пор — △: расхождение. Gemini принимает целиком; GPT уточняет, что штраф такого масштаба относится к MIPS с программной обработкой исключения, на x86/x64 невыровненное чтение почти бесплатно (кроме пересечения кеш-линий/страниц), а сохранность именно того кода не проверена. Историческая часть верна, «до сих пор замедляет» — не следует.

Хардкорный дебаггинг и инженерная культура

Отладка ОС велась в чистом ассемблере на четырех разных архитектурах

32:12: 80% времени разработчика уходило на исправление багов. Из-за отсутствия отладки на уровне исходного кода (source-level debugging) приходилось читать сырые дампы памяти. Ситуация усложнялась тем, что Windows NT разрабатывалась одновременно под процессоры Intel, MIPS, Alpha и PowerPC, требуя знания четырех разных наборов инструкций.

Фактчек: NT одновременно под Intel, MIPS, Alpha, PowerPC; Катлер пришёл из DEC — □: NT 3.1 вышла на x86/MIPS/Alpha, PowerPC добавлен в 3.51. Сравнение Катлера с Торвалдсом — □, но роль жёсткого владельца ядра описана верно.

Фактчек: Аутичный мозг способен только на одну задачу (монотропизм) — $\triangle$: монотропизм (Murray, Lesser, Lawson, 2005) — реальная, но теоретическая модель внимания, не установленный универсальный механизм. Трудности переключения и интенсивные интересы встречаются часто, но ни невозможность многозадачности, ни талант к программированию из диагноза не следуют.

Ассерты (asserts) с личным номером телефона — радикальный способ поиска багов 34:24: При поиске плавающего бага в Task Manager (загрузка ЦП >100%) разработчики ядра отказывались признавать ошибку на своей стороне. Внедрение в код ассерта с текстом «Если вы видите это, позвоните Дэйву по номеру...» привело к срабатыванию на тестовой машине с подключенным дебаггером. Это доказало, что проблема кроется в системе учета времени ядра. Номер телефона остался закомментированным в утекших исходниках Windows NT.

Ассерты должны проверять только абсолютно невозможные состояния 36:34: Использование assert для проверки предупреждений или ожидаемых ошибок — плохая практика. Ассерт должен останавливать выполнение программы только тогда, когда нарушена фундаментальная логика (например, указатель, который математически не может быть null, оказался null). Это заставляет немедленно решать архитектурную проблему.

Фактчек: Assert — только для нарушений внутренних инвариантов, не для ожидаемых ошибок — $\square$: корректный инженерный принцип. Оговорка для C/C++: `assert` исчезает при сборке с `NDEBUG`, поэтому на него нельзя вешать проверки безопасности и действия с побочными эффектами.

Создание легендарных программ: Task Manager, ZIP-папки и Pinball

Task Manager был написан дома и весил всего 87 КБ 37:05: Программа создавалась как личный сайд-проект. Для обеспечения мгновенного отклика использовались внутренние недокументированные API (NtQuerySystemInformation). Чтобы минимизировать размер, программа не линковалась с библиотеками времени выполнения C (C runtimes), что сэкономило 96 КБ. Конструкторы C++ объектов вызывались вручную через таблицы диспетчеризации.

Фактчек: ИИ ускоряет в 4 раза, но остаётся «продвинутым автокомплитом», production с нуля невозможен — $\triangle$: универсального коэффициента нет (в эксперименте METR 2025 опытные разработчики на знакомых репозиториях даже замедлялись); при этом тезис про «автокомплит» уже устарел — выпуск записан до широкого распространения агентных режимов (Claude Code, Codex). Ядро верно: выигрывает тот, кто умеет читать код и знает, что запрашивать.

Отрисовка Task Manager использует алгоритм, похожий на код Хэмминга 40:13: Для плавной работы интерфейса (даже при ресайзе окна с 32 графиками

ЦП) каждый столбец и строка имеют бит «грязного» состояния. Программа анализирует пересечения X и Y, чтобы перерисовывать только те конкретные ячейки, значения в которых изменились, не тратя ресурсы на обновление всего экрана.

Физика в Windows-версии Pinball отличается от оригинала из-за частоты кадров

42:19: Портирование Space Cadet Pinball с Win95 на NT требовало переписывания огромного конечного автомата (state engine на 50 состояний) с ассемблера x86 на C. В современной версии физика шарика рассчитывается иначе: игра пытается отрисовать максимум кадров, и на современных ПК физический цикл интерполируется до 5000 раз в секунду вместо оригинальных 30 раз.

Нативная поддержка ZIP-архивов была куплена Microsoft у собственного сотрудника

53:37: Желание купить дом и Corvette мотивировало написать shell-расширение Visual ZIP для работы с архивами как с обычными папками.

Программа продавалась как shareware. Представители Microsoft позвонили автору с предложением о покупке, не зная, что он уже работает в соседнем здании. Из финальной версии для Windows пришлось вырезать шифрование, так как в 90-е криптография приравнивалась к экспорту оружия.

Активация Windows, интерфейсы и маркетинг

Система активации Windows XP была написана в экстремальной спешке

45:27: Команда, отвечавшая за перенос кода активации из Microsoft Office в Windows, сорвала сроки. Систему пришлось создавать с нуля перед самым релизом. Самым болезненным элементом стала активация по телефону: пользователям приходилось диктовать длинные хеши, привязанные к объему памяти и жесткого диска, что превращало работу операторов колл-центра в пытку.

Удаление кастомизации интерфейса (например, перемещения панели задач) продиктовано сроками

48:10: В Windows 11 убрали возможность ставить панель задач сбоку или сверху. Поддержка таких функций экспоненциально увеличивает сложность кода отрисовки и площадь для багов. Отказ от кастомизации позволяет выпустить продукт на месяцы раньше, но разрушает лояльность power-юзеров и лишает ОС статуса «инструмента для гиков».

Агрессивный апселл и навязывание настроек разрушают доверие к ОС

1:08:31: Опыт управления собственной компанией Software Online (которая получила иск от генпрокурора за ежедневные попапы после окончания trial-периода и автоматическое включение платной доставки CD-диска) показал: навязывание действий вызывает отторжение. Современная тактика Windows с кнопкой «Использовать рекомендованные настройки» (которая сбрасывает

браузер на Edge) воспринимается пользователями как враждебное поведение системы.

Архитектура систем, синие экраны и старое железо

Синий экран смерти (BSOD) — это механизм защиты от разрушения данных 57:53: Когда ядро ОС попадает в нелегальное состояние (например, драйвер пытается дважды освободить один и тот же участок памяти), система немедленно останавливается, чтобы предотвратить запись поврежденных данных на диск. Цветовая схема (белый текст на синем фоне) была выбрана автором Джоном Виртом просто потому, что в таких же цветах работала прошивка MIPS и его редактор кода Visual SlickEdit.

Фактчек: Цвета BSOD выбрал «Джон Вирт» из-за прошивки MIPS и SlickEdit — [] по сути, [] по имени: это John **Vert**, не Wirth (ошибка транслитерации пересказа). Сам Верт подтверждал историю через блог Реймонда Чена.

Перезагрузка решает 90% проблем из-за утечек памяти и редких состояний 59:25: Со временем программы не освобождают выделенную память (утечки в куче или файле подкачки), а сложный софт попадает в состояния, которые разработчики не тестировали. Перезагрузка возвращает систему в чистое, детерминированное состояние без накопленных ошибок.

Игровые автоматы (слоты) не используют физику вращения барабанов 1:17:26: Внутри слота работает генератор случайных чисел, который мгновенно определяет математический исход (выигрыш или проигрыш) и размер выплаты. Только после этого программа генерирует визуальный «фильм» с вращением барабанов, подгоняя картинку под уже известный результат.

Восстановление мейнфреймов PDP-11 требует переписывания ядра ОС 1:19:30: В качестве хобби собираются максимальные комплектации старых ЭВМ (например, PDP-11/83 с 4 МБ памяти). Добавление любого нового контроллера диска в такие системы невозможно через простую установку драйвера — требуется полная перекомпиляция ядра BSD с интеграцией нового исходного кода.

Фактчек: PDP-11 — «мейнфрейм» — []: PDP-11 — классический миникомпьютер, мейнфреймы — IBM System/360 и подобные. Про 4 МБ в 11/83 (22-битная адресация) и пересборку ядра 2.11BSD под новый контроллер — [].

Жизнь и работа с аутизмом

Аутичный мозг работает по принципу монотропизма 1:21:34: В отличие от нейротипичных людей, мозг при РАС способен выполнять только одну задачу, но с экстремальной интенсивностью и фокусом. Многозадачность практически невозможна. Главное преимущество — способность направить невероятный

объем концентрации на решение сложной инженерной проблемы (если она вызывает интерес).

Фактчек: Meltdown отключает лобную долю, управление берёт «рептильный мозг» — Δ : расхождение по строгости (Claude Δ , GPT $\square$), но по сути согласие. При остром стрессе префронтальная регуляция снижается, а не «выключается»; «триединый мозг» Маклина отвергнут нейронаукой; сенсорная перегрузка и meltdown — не синонимы. Как метафора работает, как механизм — нет.

Социальные взаимодействия требуют запуска внутреннего «NPC-эмулятора» 1:23:09: Люди с аутизмом не считывают интуитивно чужие эмоции, сарказм или подтекст (особенно по телефону, где нет мимики). Приходится сознательно вычислять: «Как бы я себя чувствовал на месте этого человека?». Это приводит к буквальному восприятию слов и социальным ошибкам (например, публичное исправление Билла Гейтса в мелкой неточности сроков стажировки).

Маскировка (Masking) — это изнурительный процесс симуляции нормальности 1:32:47: Контроль позы, зрительного контакта, тона голоса и жестикуляции, который у обычных людей происходит автоматически, для человека в спектре является тяжелой сознательной работой.

Сенсорная перегрузка (Meltdown) отключает лобную долю мозга 1:34:22: Когда уровень стресса, фрустрации или сенсорного шума превышает предел, происходит срыв. Лобная кора, отвечающая за фильтрацию поведения и принятие решений, отключается. Управление переходит к рептильному мозгу, запускающему паническую реакцию «бей или беги».

Эмпатия при аутизме существует, но требует явной коммуникации 1:31:13: Миф об отсутствии эмпатии при РАС неверен. Проблема заключается в канале передачи данных: аутичный человек не может догадаться о чужой боли по намекам. Но если прямо и специфично озвучить проблему, уровень сопереживания будет максимальным. В отношениях это требует постоянных явных проверок статуса (пинг-понг вопросами «У тебя все хорошо?»).

Совет для карьеры: продавайте свои навыки, а не личность 1:25:52: На собеседованиях людям с РАС не стоит пытаться очаровать работодателя харизмой. Гораздо эффективнее принести портфолио, историю коммитов на GitHub и конкретные алгоритмы. При переходе на менеджерские позиции необходимо напрямую спрашивать подчиненных об их мотивации, так как интуитивно угадать чужие карьерные амбиции невозможно.

Современное программирование, ИИ и проект GitHub Primes

Проект GitHub Primes измеряет реальную скорость 100+ языков программирования 1:38:36: На сервере с 1 ТБ оперативной памяти каждую

ночь запускаются Docker-контейнеры с реализацией алгоритма «Решето Эратосфена» (поиск простых чисел до 100 млн). Правила жесткие: алгоритм должен быть идентичным, запрещено использовать более 1 бита на число, цикл длится 5 секунд на всех ядрах. На данный момент лидером по скорости чаще всего является язык Zig, а классический C отстает от него примерно в 1.5 раза.

Фактчек: GitHub Primes: Zig лидирует, C отстаёт в ~1,5 раза — $\triangle$: расхождение. Gemini подтверждает, Claude и GPT возражают: таблица меняется каждую ночь, в топе чередуются Zig/Rust/C++/C, и сравниваются конкретные реализации с конкретными флагами и категориями (потoki, битность), а не языки. Множитель 1,5 без указания даты таблицы не подтверждён.

Обучение ИИ игре в Atari Tempest требует связки ассемблера, Lua и

Python 1:04:50: Для создания ИИ, способного побить мировой рекорд в сложной векторной игре 1980 года (360 градусов движения, 11 типов врагов), ROM-файлы игры были отреверс-инжинирены. Скрипт на Lua извлекает данные из памяти эмулятора и по сокетам передает их в Python, где нейросеть (dueling Deep-Q) обучается с подкреплением. ИИ уже дошел до 36 уровня (предел человека — 96).

Фактчек: Tempest AI: MAME + Lua читает память эмулятора → сокет → Python, dueling Deep-Q — $\square$: подтверждено по репозиторию Пламмера.

Vibe-coding с помощью ИИ делает опытных инженеров

сверхпродуктивными 1:44:53: Использование LLM (например, в Cursor) позволяет писать код на незнакомых языках (Python) в 4 раза быстрее. Однако ИИ пока работает лишь как продвинутый автокомплит: без фундаментального понимания архитектуры, умения читать код и знания того, что именно нужно запросить, создать работающий production-продукт с нуля невозможно.

Будущее программирования похоже на эволюцию AutoCAD 1:46:28:

Разработка перейдет от написания отдельных строк кода к высокоуровневому проектированию. Программист будет оперировать крупными блоками и интерфейсами, описывая логику их взаимодействия для ИИ, подобно тому, как современные архитекторы собирают мосты из готовых секций в САПР, а не сваривают балки вручную.

Фактчек: Прочее: криптография как экспорт оружия (ITAR до 1996, послабление в 2000); RNG определяет исход слота до анимации (в т.ч. в механических «степперах»); Win11 убрала боковую панель задач; иск генпрокурора Вашингтона к Software Online (2006, scareware-попапы) — $\square$.

[01:49:32-01:50:28] Смолток, благодарности, реклама спонсоров и цитата Бьёрна Страуструпа.

ЛИЧНО ДЛЯ ТЕБЯ

Инварианты как фильтр AI-кода: заставляй модель явно прописывать ассертами состояния, которые математически невозможны, и проверяй, что они действительно невозможны — это ловит архитектурную ложь быстрее ревью. Измеряй время до проверенного результата, а не до появления кода: реализация и тест, написанные по одной неверной предпосылке, прекрасно согласуются друг с другом.

Твой флоу — целевой стандарт, но навык надо смещать выше: метафора AutoCAD означает, что ценность уходит в проектирование интерфейсов между крупными блоками. Сознательно прокачивай именно это (границы модулей, контракты, потоки данных), а не чтение синтаксиса.

Спрашивай прямо вместо угадывания: «у тебя всё ок?», «чего ты хочешь от этой работы?» — рабочее правило не только при РАС. Прикинь, сколько раз за месяц ты реально спросил у людей о мотивации, а сколько додумал. То же в отношениях.

Большие провалы — сумма мелких «не сегодня»: история отчисления Пламмера — фильтр для дрейфующих привычек (сон, тренировки, стройка). Плюс: гиперфокус и усталость от общения — не повод ставить себе диагноз; блоки сосредоточенной работы и защита от переключений полезны без привязки к нему.

КОНСЕНСУС РЕЦЕНЗЕНТОВ

Сошлись: авторство программ, история BSOD (с поправкой имени Vert), A20/HMA, четыре архитектуры NT, экспорт криптографии, слоты, принцип ассертов и ценность фокуса/прямой коммуникации. Разошлись: Gemini принимает бенчмарк Primes и «невыровненный доступ до сих пор замедляет Windows» как факты, Claude и GPT считают оба тезиса переупрощением; по meltdown/«рептильному мозгу» Claude ставит △ (метафора), GPT — □ (устаревшая модель); Gemini нейробиологические утверждения вообще не проверял.