

Der IOCCC

Wie man mit schlechtem Code gewinnt

Volker Diels-Grabsch

14. Januar 2020

Übersicht

Einleitung

Einsicht

Aussicht

Übersicht

Einleitung

Einsicht

Aussicht

Geschichte

- seit 1984
 - einer der ältesten Wettbewerbe im Netz
 - Usenet: net.lang.c
 - Jury damals: Landon Curt Noll, Larry Bassel
 - heute: Landon Curt Noll, Leonid A. Broukhis, Simon Cooper
 - 26. Runde: 2019

Geschichte

- seit 1984
- einer der ältesten Wettbewerbe im Netz
- Usenet: net.lang.c
- Jury damals: Landon Curt Noll, Larry Bassel
- heute: Landon Curt Noll, Leonid A. Broukhis, Simon Cooper
- 26. Runde: 2019

Geschichte

- seit 1984
- einer der ältesten Wettbewerbe im Netz
- Usenet: net.lang.c
- Jury damals: Landon Curt Noll, Larry Bassel
- heute: Landon Curt Noll, Leonid A. Broukhis, Simon Cooper
- 26. Runde: 2019

Geschichte

- seit 1984
- einer der ältesten Wettbewerbe im Netz
- Usenet: net.lang.c
- Jury damals: Landon Curt Noll, Larry Bassel
- heute: Landon Curt Noll, Leonid A. Broukhis, Simon Cooper
- 26. Runde: 2019

Geschichte

- seit 1984
- einer der ältesten Wettbewerbe im Netz
- Usenet: net.lang.c
- Jury damals: Landon Curt Noll, Larry Bassel
- heute: Landon Curt Noll, Leonid A. Broukhis, Simon Cooper
- 26. Runde: 2019

Geschichte

- seit 1984
- einer der ältesten Wettbewerbe im Netz
- Usenet: net.lang.c
- Jury damals: Landon Curt Noll, Larry Bassel
- heute: Landon Curt Noll, Leonid A. Broukhis, Simon Cooper
- 26. Runde: 2019

Ziele

- Unverständlichkeit von C-Programmen auf die Spitze treiben
- Trotzdem Regeln beachten!
- Demonstrieren, wie wichtig guter Programmierstil ist.
- C-Compiler stressen!
- Die subtilen Ecken von C illustrieren
- Ein sicheres Forum für schlechten C-Code.
- ⇒ Müllhalde für C-Code? Nein!

Ziele

- Unverständlichkeit von C-Programmen auf die Spitze treiben
- Trotzdem Regeln beachten!
- Demonstrieren, wie wichtig guter Programmierstil ist.
- C-Compiler stressen!
- Die subtilen Ecken von C illustrieren
- Ein sicheres Forum für schlechten C-Code.
- ⇒ Müllhalde für C-Code? Nein!

Ziele

- Unverständlichkeit von C-Programmen auf die Spitze treiben
- Trotzdem Regeln beachten!
- Demonstrieren, wie wichtig guter Programmierstil ist.
- C-Compiler stressen!
- Die subtilen Ecken von C illustrieren
- Ein sicheres Forum für schlechten C-Code.
- ⇒ Müllhalde für C-Code? Nein!

Ziele

- Unverständlichkeit von C-Programmen auf die Spitze treiben
- Trotzdem Regeln beachten!
- Demonstrieren, wie wichtig guter Programmierstil ist.
- C-Compiler stressen!
 - Die subtilen Ecken von C illustrieren
 - Ein sicheres Forum für schlechten C-Code.
 - ⇒ Müllhalde für C-Code? Nein!

Ziele

- Unverständlichkeit von C-Programmen auf die Spitze treiben
- Trotzdem Regeln beachten!
- Demonstrieren, wie wichtig guter Programmierstil ist.
- C-Compiler stressen!
- Die subtilen Ecken von C illustrieren
- Ein sicheres Forum für schlechten C-Code.
- ⇒ Müllhalde für C-Code? Nein!

Ziele

- Unverständlichkeit von C-Programmen auf die Spitze treiben
- Trotzdem Regeln beachten!
- Demonstrieren, wie wichtig guter Programmierstil ist.
- C-Compiler stressen!
- Die subtilen Ecken von C illustrieren
- Ein sicheres Forum für schlechten C-Code.
- ⇒ Müllhalde für C-Code? Nein!

Ziele

- Unverständlichkeit von C-Programmen auf die Spitze treiben
- Trotzdem Regeln beachten!
- Demonstrieren, wie wichtig guter Programmierstil ist.
- C-Compiler stressen!
- Die subtilen Ecken von C illustrieren
- Ein sicheres Forum für schlechten C-Code.
- ⇒ Müllhalde für C-Code? Nein!

Qualitäten

- Verschleierungstechniken sind eine Kunst
- Unterliegen Mode-Erscheinungen
- Programme sollen Sinnvolles oder Interessantes tun

Qualitäten

- Verschleierungstechniken sind eine Kunst
- Unterliegen Mode-Erscheinungen
- Programme sollen Sinnvolles oder Interessantes tun
 - Verkomplizierte „Hello World!“ haben es schwer
 - Intelligente Größenbeschränkung schützt vor Nadel-Heuhaufen-Taktik
 - fehlerfrei
 - portabel:
 - `gcc -std=c99 -pedantic -Werror -Wextra -Weverything -std=c99`
 - `clang -std=c99 -pedantic -Werror -Wextra -Weverything -std=c99`
 - `clang++ -std=c++98 -pedantic -Werror -Wextra -Weverything -std=c++98`
 - möglichst keine Compiler-Warnungen: `-Werror -Wall -Wextra -Weverything -pedantic -std=c99`
 - eigene Subkultur
 - viele Insider-Scherze

Qualitäten

- Verschleierungstechniken sind eine Kunst
- Unterliegen Mode-Erscheinungen
- Programme sollen Sinnvolles oder Interessantes tun
 - Verkomplizierte „Hello World!“ haben es schwer
 - Intelligente Größenbeschränkung schützt vor Nadel-Heuhaufen-Taktik
 - fehlerfrei
 - portabel:
 - `gcc -std=c99 -pedantic -Werror -Wall -Wextra -Weverything -std=c99`
 - möglichst keine Compiler-Warnungen: -Werror -Wall -Wextra -Weverything -pedantic -std=c99
 - eigene Subkultur
 - viele Insider-Scherze

Qualitäten

- Verschleierungstechniken sind eine Kunst
 - Unterliegen Mode-Erscheinungen
 - Programme sollen Sinnvolles oder Interessantes tun
 - Verkomplizierte „Hello World!“ haben es schwer
 - Intelligente Größenbeschränkung schützt vor Nadel-Heuhaufen-Taktik
 - fehlerfrei
 - portabel:
-
- möglichst keine Compiler-Warnungen: -Werror -Wall -Wextra -Weverything -pedantic -std=c99
 - eigene Subkultur
 - viele Insider-Scherze

Qualitäten

- Verschleierungstechniken sind eine Kunst
- Unterliegen Mode-Erscheinungen
- Programme sollen Sinnvolles oder Interessantes tun
 - Verkomplizierte „Hello World!“ haben es schwer
 - Intelligente Größenbeschränkung schützt vor Nadel-Heuhaufen-Taktik
 - fehlerfrei
 - portabel:
 - mindestens GCC + Qiang
 - kein Systemcall (z.B. malloc)
 - kein Systemcall (z.B. fopen)
 - möglichst keine Compiler-Warnungen: -Werror -Wall -Wextra -Weverything -pedantic -std=c99
 - eigene Subkultur
 - viele Insider-Scherze

Qualitäten

- Verschleierungstechniken sind eine Kunst
- Unterliegen Mode-Erscheinungen
- Programme sollen Sinnvolles oder Interessantes tun
 - Verkomplizierte „Hello World!“ haben es schwer
 - Intelligente Größenbeschränkung schützt vor Nadel-Heuhaufen-Taktik
 - fehlerfrei
 - portabel:
 - mindestens GCC + Clang
 - mindestens 32 + 64 Bit
 - (ältere+neuere Compiler)
 - möglichst keine Compiler-Warnungen: -Werror -Wall -Wextra -Weverything -pedantic -std=c99
 - eigene Subkultur
 - viele Insider-Scherze

Qualitäten

- Verschleierungstechniken sind eine Kunst
- Unterliegen Mode-Erscheinungen
- Programme sollen Sinnvolles oder Interessantes tun
 - Verkomplizierte „Hello World!“ haben es schwer
 - Intelligente Größenbeschränkung schützt vor Nadel-Heuhaufen-Taktik
 - fehlerfrei
 - portabel:
 - mindestens GCC + Clang
 - mindestens 32 + 64 Bit
 - (ältere+neuere Compiler)
 - möglichst keine Compiler-Warnungen: -Werror -Wall -Wextra -Weverything -pedantic -std=c99
 - eigene Subkultur
 - viele Insider-Scherze

Qualitäten

- Verschleierungstechniken sind eine Kunst
- Unterliegen Mode-Erscheinungen
- Programme sollen Sinnvolles oder Interessantes tun
 - Verkomplizierte „Hello World!“ haben es schwer
 - Intelligente Größenbeschränkung schützt vor Nadel-Heuhaufen-Taktik
 - fehlerfrei
 - portabel:
 - mindestens GCC + Clang
 - mindestens 32 + 64 Bit
 - (ältere+neuere Compiler)
 - möglichst keine Compiler-Warnungen: -Werror -Wall -Wextra -Weverything -pedantic -std=c99
 - eigene Subkultur
 - viele Insider-Scherze

Qualitäten

- Verschleierungstechniken sind eine Kunst
- Unterliegen Mode-Erscheinungen
- Programme sollen Sinnvolles oder Interessantes tun
 - Verkomplizierte „Hello World!“ haben es schwer
 - Intelligente Größenbeschränkung schützt vor Nadel-Heuhaufen-Taktik
 - fehlerfrei
 - portabel:
 - mindestens GCC + Clang
 - mindestens 32 + 64 Bit
 - (ältere+neuere Compiler)
- möglichst keine Compiler-Warnungen: -Werror -Wall -Wextra -Weverything -pedantic -std=c99
- eigene Subkultur
- viele Insider-Scherze

Qualitäten

- Verschleierungstechniken sind eine Kunst
- Unterliegen Mode-Erscheinungen
- Programme sollen Sinnvolles oder Interessantes tun
 - Verkomplizierte „Hello World!“ haben es schwer
 - Intelligente Größenbeschränkung schützt vor Nadel-Heuhaufen-Taktik
 - fehlerfrei
 - portabel:
 - mindestens GCC + Clang
 - mindestens 32 + 64 Bit
 - (ältere+neuere Compiler)
 - möglichst keine Compiler-Warnungen: -Werror -Wall -Wextra -Weverything -pedantic -std=c99
 - eigene Subkultur
 - viele Insider-Scherze

Qualitäten

- Verschleierungstechniken sind eine Kunst
- Unterliegen Mode-Erscheinungen
- Programme sollen Sinnvolles oder Interessantes tun
 - Verkomplizierte „Hello World!“ haben es schwer
 - Intelligente Größenbeschränkung schützt vor Nadel-Heuhaufen-Taktik
 - fehlerfrei
 - portabel:
 - mindestens GCC + Clang
 - mindestens 32 + 64 Bit
 - (ältere+neuere Compiler)
 - möglichst keine Compiler-Warnungen: -Werror -Wall -Wextra -Weverything -pedantic -std=c99
 - eigene Subkultur
 - viele Insider-Scherze

Qualitäten

- Verschleierungstechniken sind eine Kunst
- Unterliegen Mode-Erscheinungen
- Programme sollen Sinnvolles oder Interessantes tun
 - Verkomplizierte „Hello World!“ haben es schwer
 - Intelligente Größenbeschränkung schützt vor Nadel-Heuhaufen-Taktik
 - fehlerfrei
 - portabel:
 - mindestens GCC + Clang
 - mindestens 32 + 64 Bit
 - (ältere+neuere Compiler)
 - möglichst keine Compiler-Warnungen: -Werror -Wall -Wextra -Weverything -pedantic -std=c99
 - eigene Subkultur
 - viele Insider-Scherze

Regeln

- 26 Regeln

- viele selbstironisch

- einige Formalitäten

- einige wichtig:

- Programmgröße ≤ 4000 Bytes
- Zeilenanzahl ≤ 2053
- keine veränderbaren Daten im Speicher
- keine Deadlocks
- Rückmeldungen per Forum ≤ 10 pro Tag
- zweifelhafte, automatische Tests
- kein `printf`, `scanf`, kein `TM`
- Quelltexten erlaubt, aber kein Code und ≤ 1 Map

Regeln

- 26 Regeln
- viele selbstironisch
- einige Formalitäten
- einige wichtig:

• Programmgröße ≤ 1000 Zeilen

• Programm ≤ 2053

• keine verbotenen Zeichen (z.B. `++`)

• keine `typedef`

• keine Funktionen mit Parameter `...`

• keine `enum`, automatische Typen

• keine `goto`, `sizeof`, `union`

• Funktionsaufrufe erlaubt, aber kein Code und `_` -Name

Regeln

- 26 Regeln
- viele selbstironisch
- einige Formalitäten
- einige wichtig:

- Programmgröße ≤ 4096 Bytes

- $\text{max}(\text{code}) \leq 255$

- keine ungenutzten Daten im Speicher

- keine Deadlocks

- Kurzkommunikation und $\text{max}(\text{code}) \leq 255$

- zweifelhafte, automatische Code

- kein $\text{max}(\text{code})$ geben, kein max

- Funktionen erlaubt, aber kein Code und $\text{max}(\text{code})$

Regeln

- 26 Regeln
- viele selbstironisch
- einige Formalitäten
- einige wichtig:
 - Programmgröße ≤ 4096 Bytes
 - `iocccsize -i` ≤ 2053
 - keine vorhandenen Dateien modifizieren
 - harte Deadline
 - Einreichungen pro Runde $\leq 8,000000$
 - zuverlässiger, automatischer Build
 - kein 8-bit-Zeichen, kein \hat{M}
 - Zusatzdateien erlaubt, aber kein Code und ≤ 1 MiB

Regeln

- 26 Regeln
- viele selbstironisch
- einige Formalitäten
- einige wichtig:
 - Programmgröße ≤ 4096 Bytes
 - `iocccsize -i ≤ 2053`
 - keine vorhandenen Dateien modifizieren
 - harte Deadline
 - Einreichungen pro Runde $\leq 8,000000$
 - zuverlässiger, automatischer Build
 - kein 8-bit-Zeichen, kein \hat{M}
 - Zusatzdateien erlaubt, aber kein Code und ≤ 1 MiB

Regeln

- 26 Regeln
- viele selbstironisch
- einige Formalitäten
- einige wichtig:
 - Programmgröße ≤ 4096 Bytes
 - `iocccsize -i ≤ 2053`
 - keine vorhandenen Dateien modifizieren
 - harte Deadline
 - Einreichungen pro Runde $\leq 8,000000$
 - zuverlässiger, automatischer Build
 - kein 8-bit-Zeichen, kein \hat{M}
 - Zusatzdateien erlaubt, aber kein Code und ≤ 1 MiB

Regeln

- 26 Regeln
- viele selbstironisch
- einige Formalitäten
- einige wichtig:
 - Programmgröße ≤ 4096 Bytes
 - `iocccsize -i ≤ 2053`
 - keine vorhandenen Dateien modifizieren
 - harte Deadline
 - Einreichungen pro Runde $\leq 8,000000$
 - zuverlässiger, automatischer Build
 - kein 8-bit-Zeichen, kein \hat{M}
 - Zusatzdateien erlaubt, aber kein Code und ≤ 1 MiB

Regeln

- 26 Regeln
- viele selbstironisch
- einige Formalitäten
- einige wichtig:
 - Programmgröße ≤ 4096 Bytes
 - `iocccsize -i ≤ 2053`
 - keine vorhandenen Dateien modifizieren
 - harte Deadline
 - Einreichungen pro Runde $\leq 8,000000$
 - zuverlässiger, automatischer Build
 - kein 8-bit-Zeichen, kein \hat{M}
 - Zusatzdateien erlaubt, aber kein Code und ≤ 1 MiB

Regeln

- 26 Regeln
- viele selbstironisch
- einige Formalitäten
- einige wichtig:
 - Programmgröße ≤ 4096 Bytes
 - `iocccsize -i ≤ 2053`
 - keine vorhandenen Dateien modifizieren
 - harte Deadline
 - Einreichungen pro Runde $\leq 8,000000$
 - zuverlässiger, automatischer Build
 - kein 8-bit-Zeichen, kein \hat{M}
 - Zusatzdateien erlaubt, aber kein Code und ≤ 1 MiB

Regeln

- 26 Regeln
- viele selbstironisch
- einige Formalitäten
- einige wichtig:
 - Programmgröße ≤ 4096 Bytes
 - `iocccsize -i` ≤ 2053
 - keine vorhandenen Dateien modifizieren
 - harte Deadline
 - Einreichungen pro Runde $\leq 8,000000$
 - zuverlässiger, automatischer Build
 - kein 8-bit-Zeichen, kein \hat{M}
 - Zusatzdateien erlaubt, aber kein Code und ≤ 1 MiB

Regeln

- 26 Regeln
- viele selbstironisch
- einige Formalitäten
- einige wichtig:
 - Programmgröße ≤ 4096 Bytes
 - `iocccsize -i` ≤ 2053
 - keine vorhandenen Dateien modifizieren
 - harte Deadline
 - Einreichungen pro Runde $\leq 8,000000$
 - zuverlässiger, automatischer Build
 - kein 8-bit-Zeichen, kein \hat{M}
 - Zusatzdateien erlaubt, aber kein Code und ≤ 1 MiB

Regeln

- 26 Regeln
- viele selbstironisch
- einige Formalitäten
- einige wichtig:
 - Programmgröße ≤ 4096 Bytes
 - `iocccsize -i` ≤ 2053
 - keine vorhandenen Dateien modifizieren
 - harte Deadline
 - Einreichungen pro Runde $\leq 8,000000$
 - zuverlässiger, automatischer Build
 - kein 8-bit-Zeichen, kein \hat{M}
 - Zusatzdateien erlaubt, aber kein Code und ≤ 1 MiB

.

- Regeln entwickeln sich.
- Kreative Umgehung von Regeln wird belohnt.
- Aber Regeln werden verschärft.
- Beispiel: Regeln sind keine Parabolik verboten!
- ergänzt durch Richtlinien
- ergänzt durch Hinweise in Jury-Bemerkungen
- ergänzt durch Hinweise, was für Einträge es knapp nicht schafften

.

- Regeln entwickeln sich.
- Kreative Umgehung von Regeln wird belohnt.
- Aber Regeln werden verschärft.
 - Beispiel: Warum wird heute Portabilität verlangt?
- ergänzt durch Richtlinien
- ergänzt durch Hinweise in Jury-Bemerkungen
- ergänzt durch Hinweise, was für Einträge es knapp nicht schafften

.

- Regeln entwickeln sich.
- Kreative Umgehung von Regeln wird belohnt.
- Aber Regeln werden verschärft.
 - Beispiel: Warum wird heute Portabilität verlangt?
- ergänzt durch Richtlinien
- ergänzt durch Hinweise in Jury-Bemerkungen
- ergänzt durch Hinweise, was für Einträge es knapp nicht schafften

.

- Regeln entwickeln sich.
- Kreative Umgehung von Regeln wird belohnt.
- Aber Regeln werden verschärft.
 - Beispiel: Warum wird heute Portabilität verlangt?
- ergänzt durch Richtlinien
- ergänzt durch Hinweise in Jury-Bemerkungen
- ergänzt durch Hinweise, was für Einträge es knapp nicht schafften

.

- Regeln entwickeln sich.
- Kreative Umgehung von Regeln wird belohnt.
- Aber Regeln werden verschärft.
 - Beispiel: Warum wird heute Portabilität verlangt?
- ergänzt durch Richtlinien
- ergänzt durch Hinweise in Jury-Bemerkungen
- ergänzt durch Hinweise, was für Einträge es knapp nicht schafften

.

- Regeln entwickeln sich.
- Kreative Umgehung von Regeln wird belohnt.
- Aber Regeln werden verschärft.
 - Beispiel: Warum wird heute Portabilität verlangt?
- ergänzt durch Richtlinien
- ergänzt durch Hinweise in Jury-Bemerkungen
- ergänzt durch Hinweise, was für Einträge es knapp nicht schafften

.

- Regeln entwickeln sich.
- Kreative Umgehung von Regeln wird belohnt.
- Aber Regeln werden verschärft.
 - Beispiel: Warum wird heute Portabilität verlangt?
- ergänzt durch Richtlinien
- ergänzt durch Hinweise in Jury-Bemerkungen
- ergänzt durch Hinweise, was für Einträge es knapp nicht schafften

Übersicht

Einleitung

Einsicht

Aussicht

Übersicht

Einleitung

Einsicht

Aussicht

Ablauf

- 27. Runde beginnt morgen!
 - 2020-Jan-15 15:15:15 UTC
- Macht mit!
- <https://www.ioccc.org/>

Ablauf

- 27. Runde beginnt morgen!
 - 2020-Jan-15 15:15:15 UTC
- Macht mit!
- <https://www.ioccc.org/>

Ablauf

- 27. Runde beginnt morgen!
 - 2020-Jan-15 15:15:15 UTC
- Macht mit!
- <https://www.ioccc.org/>

Ablauf

- 27. Runde beginnt morgen!
 - 2020-Jan-15 15:15:15 UTC
- Macht mit!
- <https://www.ioccc.org/>

Der IOCCC

Wie man mit schlechtem Code gewinnt

Volker Diels-Grabsch



`https://njh.eu/`

