תשובות תקן ANSI-C

תשובה 1

תמיד אינו נכון,
אופן הגישה לתיקיות והפורמט שלהן משתנים באופן מהותי בין מערכות הפעלה שונות.
עמוד 179 בספר משווה בין UNIX לבין MS-DOS ומציין שבמערכות אחרות (כגון MS-DOS), ייתכן שיהיה צורך בקריאת מערכת (system call) אפילו רק כדי לגשת לשמות הקבצים. לעומת UNIX, שבה תיקייה היא פשוט קובץ שניתן לקרוא.

תשובה 2

תמיד אינו נכון,
לפי **עמוד 179 בספר**,
"במערכת UNIX תיקייה היא פשוט קובץ".
התכונות של הקובץ (מלבד שמו) נשמר במבנה נתונים הנקרא inode, והוא משותף הן לקבצים רגילים והן לתיקיות. והתכונות זהות, כאשר ההבדל היחיד הוא בערך של st_mode המציין האם מדובר בתיקייה או בקובץ רגיל.

תשובה 3

תמיד אינו נכון,
לפי **עמוד 169 בספר**,
"קיים ממשק יחיד והומוגני המטפל בכל התקשורת בין התוכנית להתקנים השונים".

תשובה 4

תמיד אינו נכון,
לפי **עמוד 173 בספר**,
ניתן לקבוע מראש את הרשאות הקובץ כבר בפקודת היצירה create באמצעות הפרמטר perms.

תשובה 5

תמיד אינו נכון,
לפי **עמוד 174 בספר**,
"קיימת מגבלה על מספר הקבצים שתוכנית יכולה לפתוח בו-זמנית ערך זה עומד לעיתים קרובות על כ-20 קבצים".

תשובה 6

תמיד נכון,
לפי **עמוד 174 בספר**,
מערכת UNIX מאפשרת לקרוא מקובץ בכל סדר שרירותי בעת הצורך. באמצעות קריאת המערכת lseek שמאפשרת להזיז את המיקום הנוכחי בקובץ ו"לשוטט" בו.

תשובה 7

תמיד אינו נכון,
לפי **עמוד 171 בספר**, קריאות מערכת הן למעשה פונקציות בתוך מערכת ההפעלה שניתן לקרוא להן מכל תוכנית משתמש דרך syscalls.h, כולל מתוך התוכנית הראשית (main). אין שום דרישה לכתוב בפונקציית ספרייה מיוחדת לצורך קריאה מהתוכנית הראשית.

תשובה 8

תמיד אינו נכון,
לפי **עמוד 188 בספר**,
"הקצאת זיכרון (storage allocation) היא תלוית-מכונה באופן מהותי".

תשובה 9

תמיד אינו נכון,
לפי **עמוד 149 בספר**,
שדה סיביות (bit-field) מוגדר כסט של סיביות סמוכות בתוך יחידת אחסון המוגדרת במימוש. ללא מגבלה על מספר השדות (כגון 8), אלא ששדות סמוכים נדחסים לתוך אותה יחידת אחסון, ואם השדה הבא לא נכנס הוא עשוי לעבור ליחידה הבאה.

תשובה 10

תמיד אינו נכון,
לפי **עמוד 150 בספר**,
לbit-field אין כתובת זיכרון רגילה כך שלא ניתן להשתמש באופרטור הכתובת & עליו.

תשובה 11

תמיד אינו נכון,
לפי **עמודים 97, 99 ו- 103 בספר**,
השימוש במצביעים לרוב יותר יעיל וקריא מאשר שימוש במערכים.
גישה למערך באמצעות אינדקס (a[i]) שקולה לגישה באמצעות מצביע ((a+i)*).
ובשפת C המהדר דואג אוטומטית להתאים את הפעולות (כגון קידום מצביע) לגודל האובייקט המוצבע בסביבה, ולכן אינו תלוי בה.

תשובה 12

תמיד אינו נכון,
לפי **עמוד 170 בספר**,
בכל קריאה של read יעיל יותר לקרוא בלוקים גדולים של בתים בכל קריאה ולא בית בודד. כך יתבצעו פחות קריאות מערכת.

תשובה 13

לפעמים נכון ולפעמים אינו נכון,
לפי **עמוד 129 בספר,**
ניתן להעביר רשומות כארגומנטים לפונקציות, בכמה דרכים נוספות. למשל העברת המבנה כולו (by value), כלומר כערך ולא רק לפי כתובת. אבל הנ"ל אינו מומלץ משיקולי מקום במחסנית.

תשובה 14

תמיד אינו נכון,
לפי **עמודים 155 ו- 254 בספר,**
ישנן פונקציות המקבלות מספר משתנה של ארגומנטים שאינו ידוע מראש, באמצעות הסימון "...". למשל הפונקציה printf, שהצהרתה התקנית מסתיימת בסימון הנ"ל. הספריה <stdarg.h> מספקת כלים למעבר על רשימת ארגומנטים שמספרם אינו ידוע.

תשובה 15

תמיד נכון,
לפי **עמוד 70 בספר,**
גם כאשר טיפוס ההחזרה של פונקציה אינו מוגדר, הפונקציה תחזיר int כברירת מחדל. ולכן בכל מקרה שבו הערך החוזר הוא void הפונקציה מחזירה ערך

תשובה 16

לפעמים נכון ולפעמים אינו נכון,
לפי **עמוד 129 בספר,**
כל פונקציה מוכרת לשאר התכנית אך ורק החל ממיקום ההצהרה שלה. ולכן יש לדאוג להצהיר על כל פונקציה לפני השימוש בה.

תשובה 17

לפעמים נכון ולפעמים אינו נכון,
לפי **עמוד 218 בספר,**
אם רשימת הפרמטרים באב-טיפוס מסתיימת בסימון "...", הפונקציה רשאית לקבל יותר ארגומנטים מאשר מספר הפרמטרים שצוינו במפורש. ולכן קיום אב-טיפוס אינו מחייב בהכרח מספר ידוע של ארגומנטים.

תשובה 18

תמיד אינו נכון,
לפי **עמוד 72 בספר,**
אם שם של פונקציה מופיע בביטוי וטרם הוצהר עליו, המהדר מצהיר עליו באופן אוטומטי. ולכן אין חובה להצהיר על אב-טיפוס לכל פונקציה כדי שהמהדר ידע על קיומה ויזהה קריאות אליה.

תשובה 19

תמיד נכון,
לפי **עמוד 26 בספר,**
הפונקציה main היא פונקציה כמו כל פונקציה אחרת. ואין שום מניעה בשפה לקרוא לה מתוך פונקציות אחרות בתוכנית או מתוך עצמה. אם כי זה מקרה פחות נפוץ.

תשובה 20

לפעמים נכון ולפעמים אינו נכון,
לפי **עמודים 26 ו- 70 בספר,**
כשפונקציה מסתיימת ללא return עם ערך, היא מחזירה שליטה לסביבה ללא ערך מוגדר והדבר נחשב תקין בשפה. אם כי מומלץ שmain תחזיר ערך לסביבה כדי לסמן הצלחה או כישלון.

תשובה 21

תמיד אינו נכון,
לפי **עמוד 83 בספר,**
ערך משתנה אוטומטי סטטי נשמר בין קריאות לפונקציה בה הוגדר. אבל "פרטי" לפונקציה. ולא נגיש מפונקציות אחרות, בניגוד למשתנה גלובלי.

תשובה 22

תמיד אינו נכון,
לפי **עמוד 85 בספר,**
אתחול (הקצאת זכרון) של משתנים חיצוניים וסטטיים מתבצע לפני שהתכנית מתחילה לרוץ. ולא כפעולה ראשונה בזמן ריצת התכנית.

תשובה 23

תמיד אינו נכון,
לפי **עמוד 31 בספר,**
אם לא מאתחלים משתנים אוטומטיים, הם מכילים ערכי "זבל". אם כי לא ניתן לעשות בהם שימוש עד שיאותחלו. ולשם תכנות נכון יש לאתחלו כבר בהגדרה.

תשובה 24, 25, 26

לפעמים נכון ולפעמים אינו נכון,
לפי **עמוד 80 בספר,**
הטווח (scope) של משתנה חיצוני מתחיל בנקודה שבה הוא הוכרז ועד סוף הקובץ שלו. אם ההגדרה של המשתנה החיצוני היא בקובץ אחר, או אחרי הגדרת הפונקציה באותו קובץ, יש להצהיר עליו שוב באמצעות extern בתוך הפונקציה.